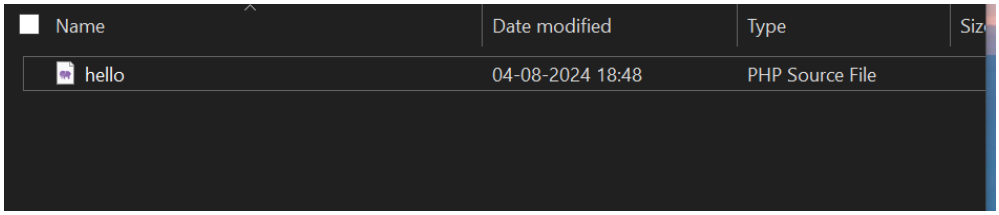
Name: Mohit Kerkar Div: D15C Roll No: 23

Adv DevOps Experiments

Exp 1a : Static Hosting

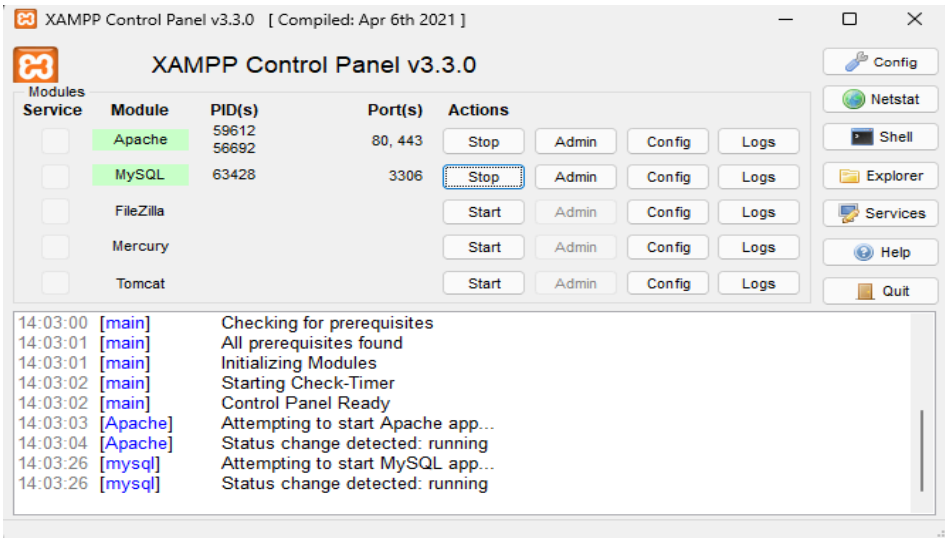
a) Hosting of a PHP file on Local virtual machine using Xampp

1. Create a .php file in some local repository

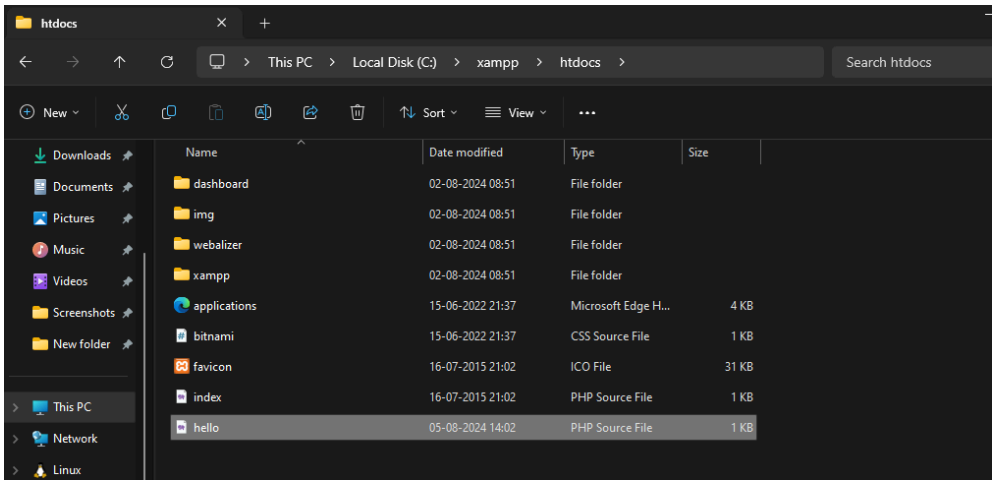


2. Make changes in that .php file as per your desire and save it

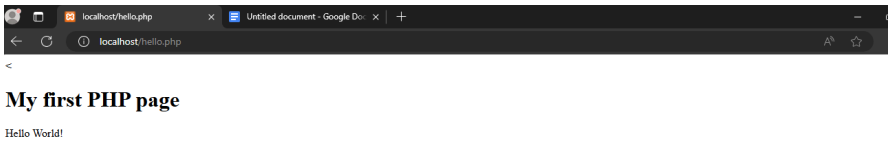
3. Make sure you have installed Xampp on your local machine. After the installation, start Xampp control panel and start modules named Apache and MySql



4. Ensure that you relocate your .php file in the htdocs folder inside the Xampp folder



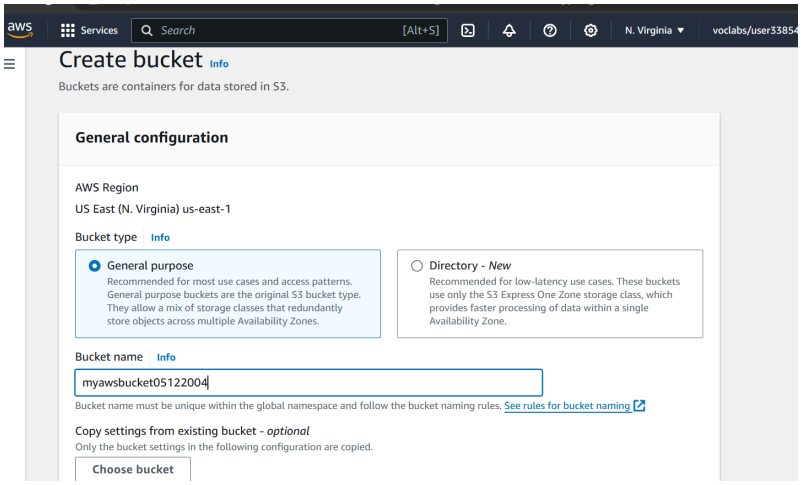
5. Access the contents of the php file by typing localhost/your_file.php on your browser. We have successfully hosted our php file on our local machine using Xampp.

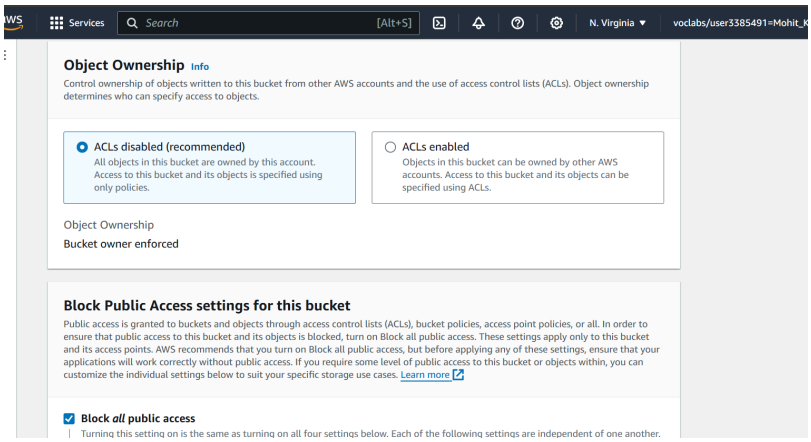


Name: Mohit Kerkar Div: D15C Roll No: 23

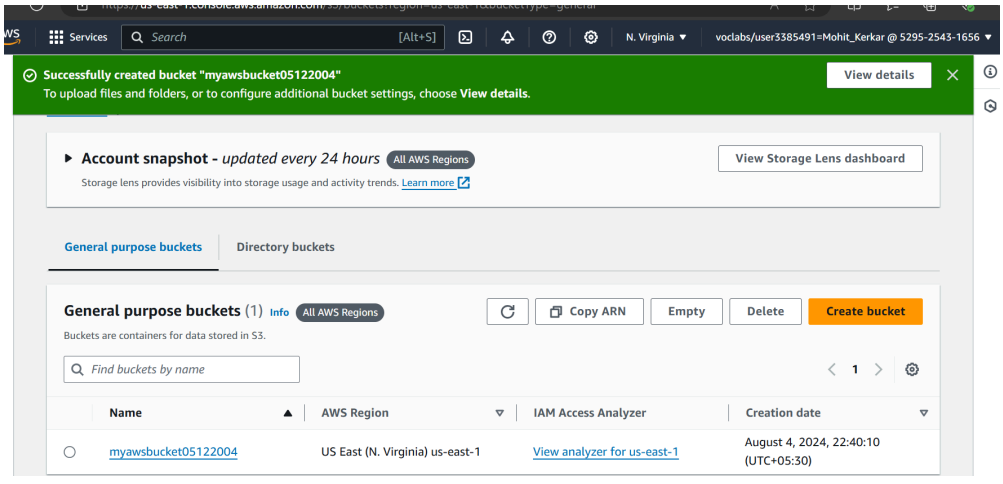
b) Static hosting using AWS S3 bucket

1. Navigate to S3 inside services and create a bucket inside S3. Select the additional settings related to the bucket.

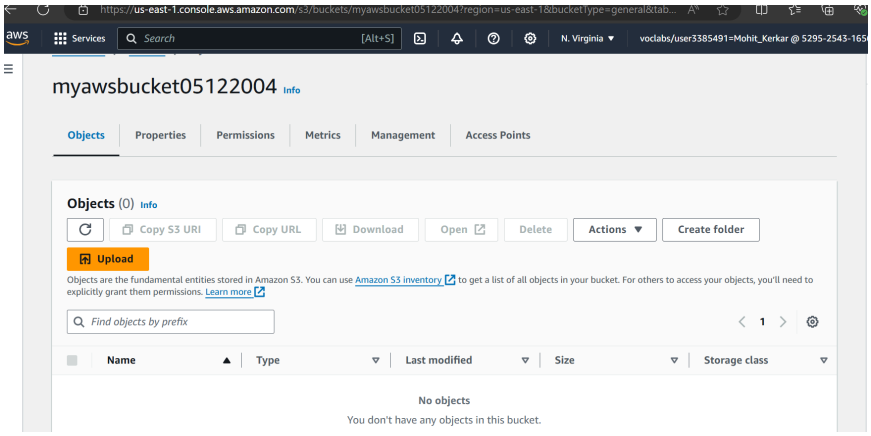


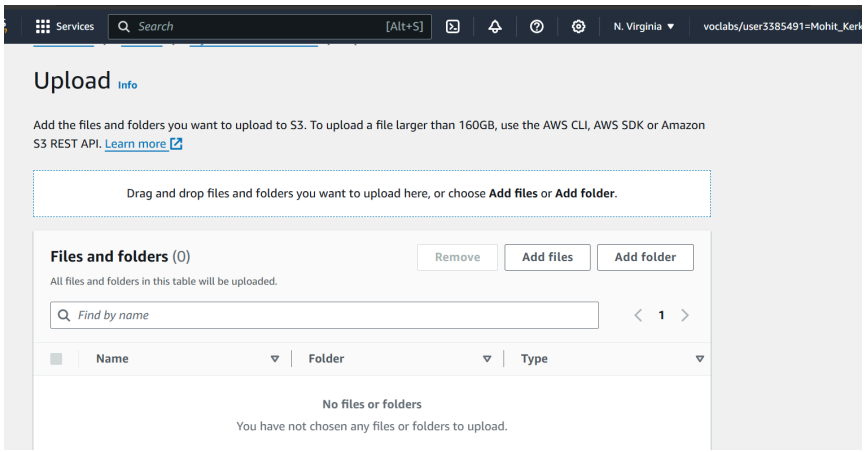


2. Our bucket was successfully created. Now, we would want to add/upload our local files onto our bucket

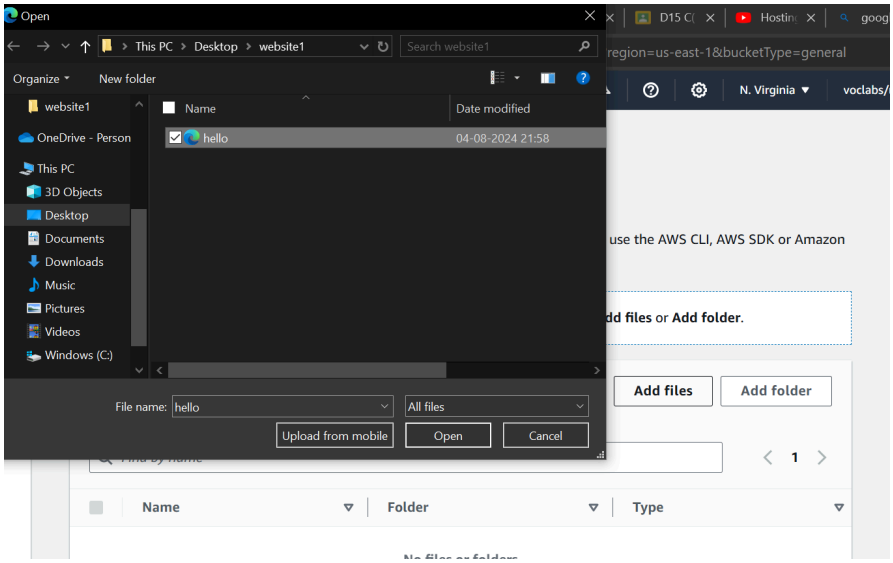


Name: Mohit Kerkar Div: D15C Roll No: 23

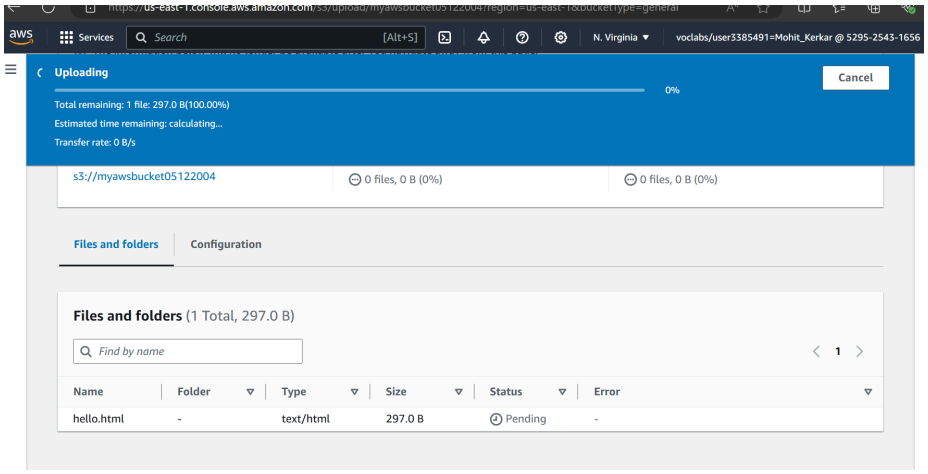




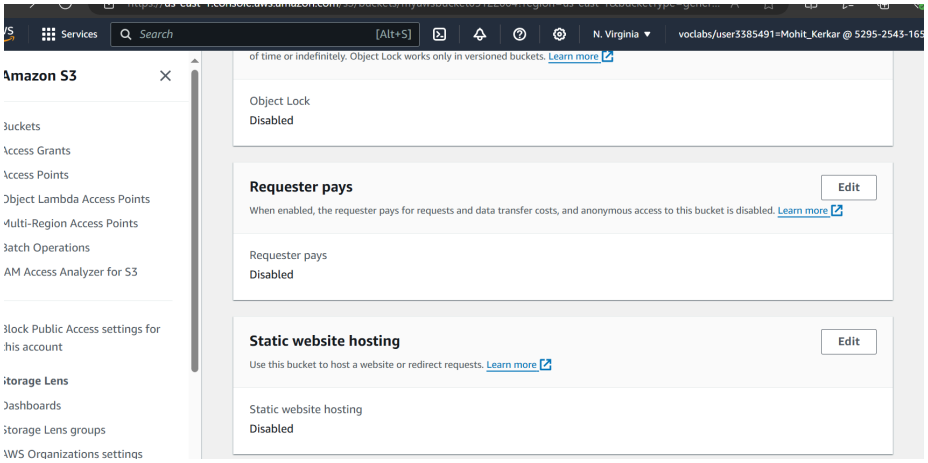
3. Create a new html file, modify it and save it with a desired file name. Now, add/upload this file onto your S3 bucket.

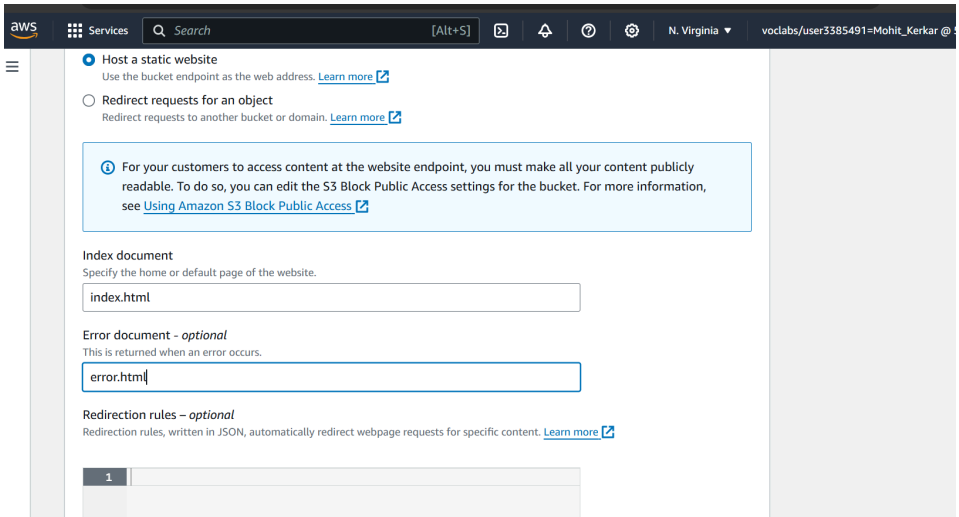


Name: Mohit Kerkar Div: D15C Roll No: 23

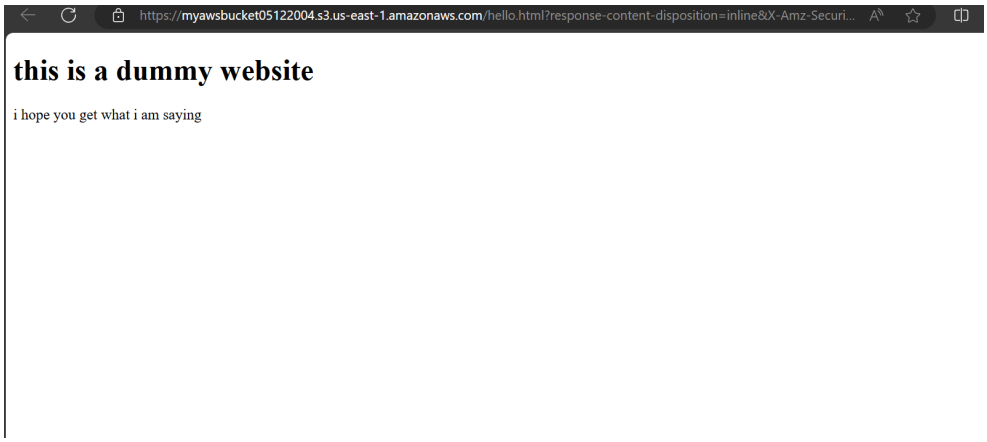


4. For enabling static website hosting, we must enable this Static website hosting option





5. Open the website using the S3 bucket. You'd be able to see the contents of you html file. Thus, we have successfully and statically hosted our html file using AWS S3 bucket



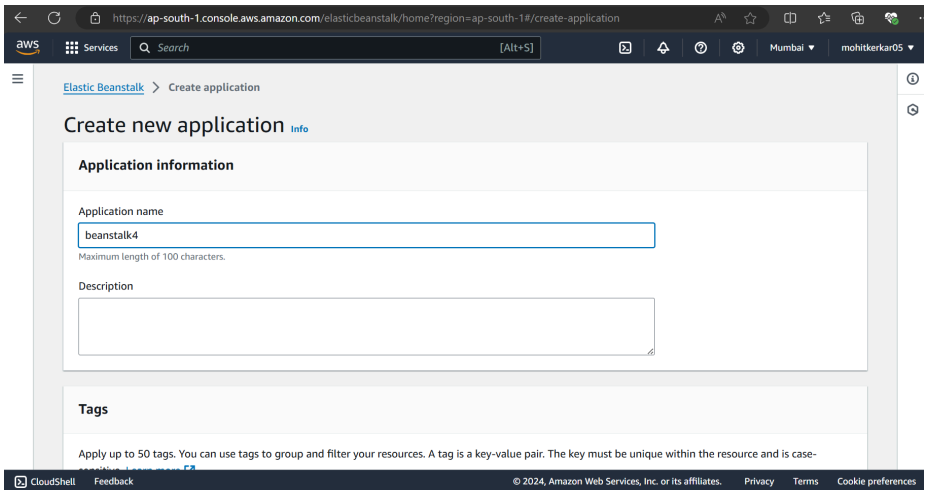
Conclusion: In conclusion, static hosting can be achieved locally using XAMPP or on AWS S3. With XAMPP, you can host PHP files on your local machine by placing them in the `htdocs` folder and accessing them via `localhost`. For cloud-based hosting, AWS S3 provides a simple solution by allowing you to upload HTML files to an S3 bucket and enable static website hosting. This approach requires minimal configuration and enables quick deployment of static content, making it ideal for small websites or testing environments.

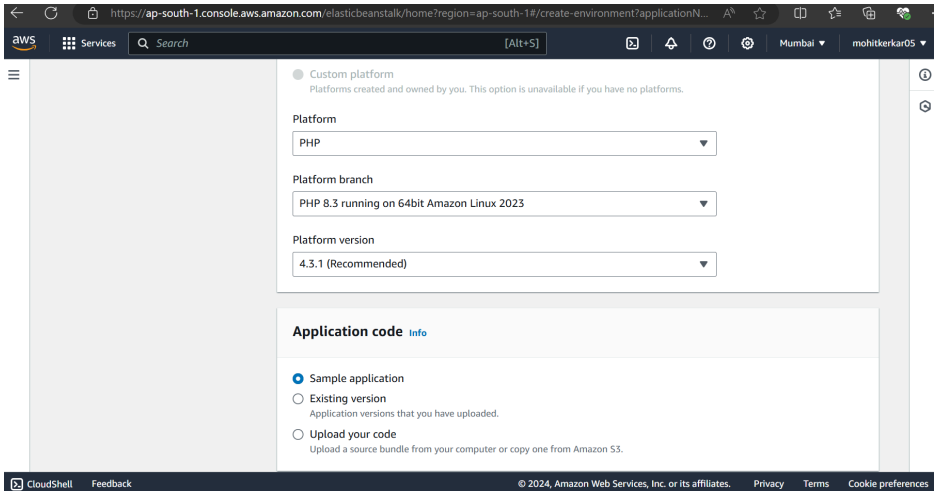
Name: Mohit Kerkar Div: D15C Roll No: 23

Exp 02:To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

Step 1: Create our ElasticBeanstalk Environment

Login into your AWS account and navigate to services. Search for Elastic Beanstalk service and click on create application. Give your application a suitable name. For the platform, select PHP. Rest of the configuration settings are to be kept as default.



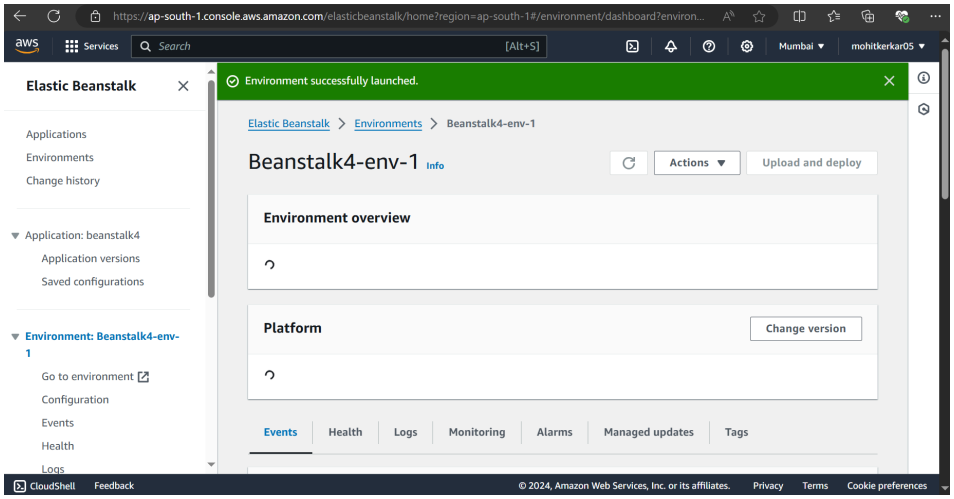


Now, while creating the environment, we are asked to provide an IAM role with the necessary EC2 permissions. We are supposed to make sure that we have made an existing IAM role with the following set of permissions:

1. AWSElasticBeanStalkWebTier
2. AWSElasticBeanStalkWorkerTier
3. AWSElasticBeanStalkMulticontainerDocker

Name: Mohit Kerkar Div: D15C Roll No: 23

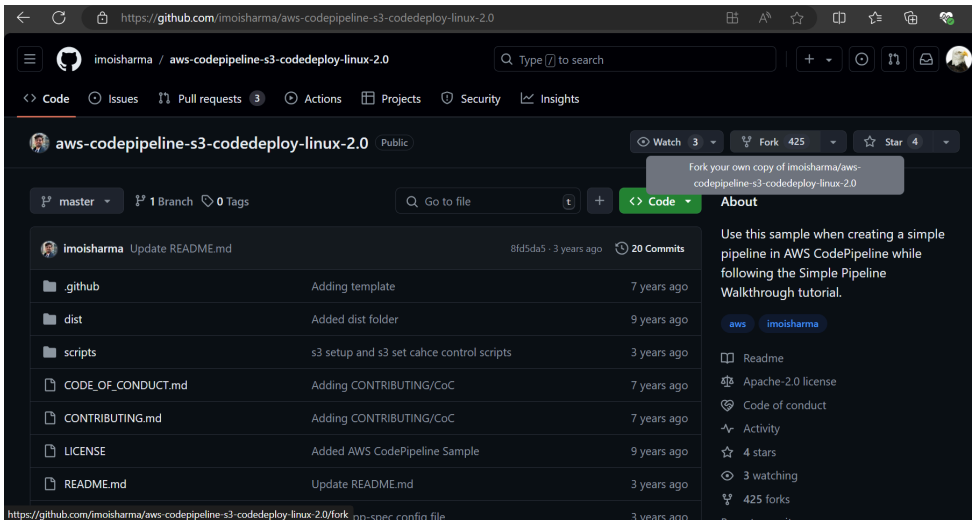
We can skip the steps to follow after the initial few steps mentioned above and move straight to review the settings of our environment. After reviewing everything properly, our environment can successfully be created.



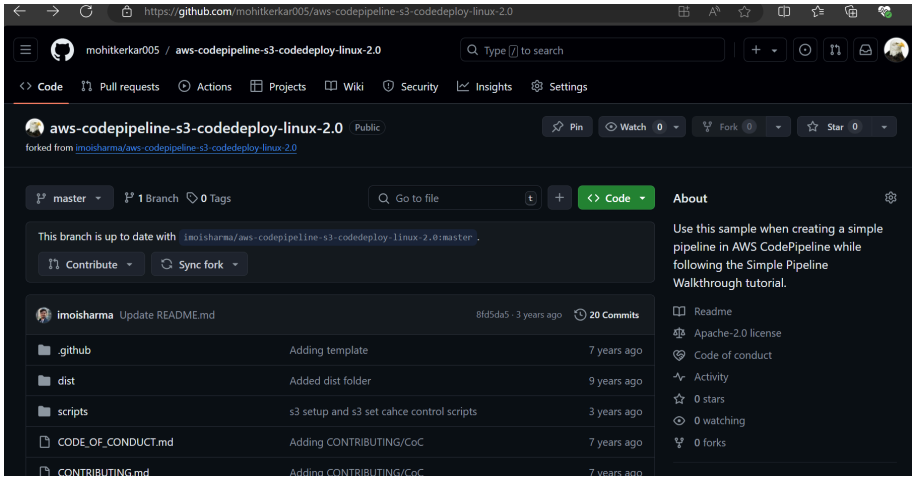
Step 2: Fork the required repository onto our github account

The repository to be forked is - imoisharma/aws-codepipeline-s3-codedeploy-linux-2.0

This step is necessary for the execution of the steps to follow. It will be helpful in the creation of a pipeline.

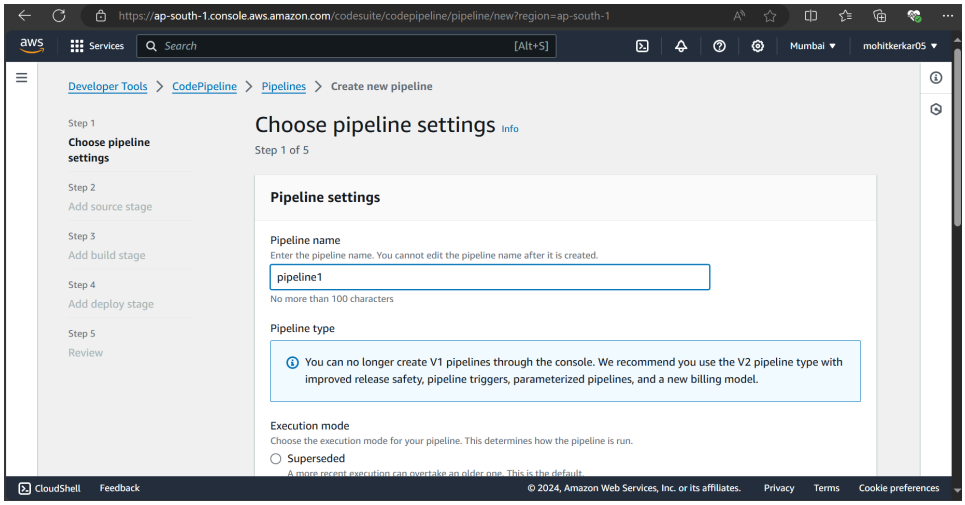


Name: Mohit Kerkar Div: D15C Roll No: 23

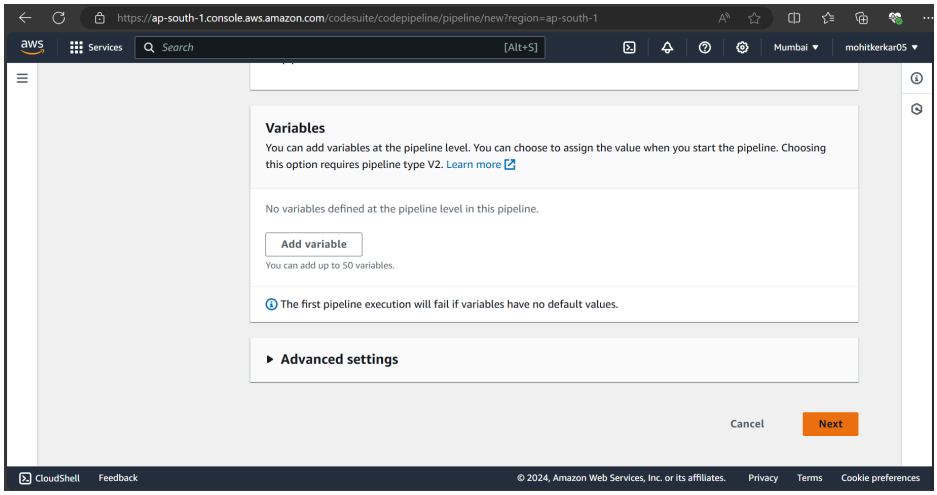


Step 3: Creation of the Pipeline

Navigate to Codepipeline inside Developer Tools. Give a suitable name to the pipeline you want to create.



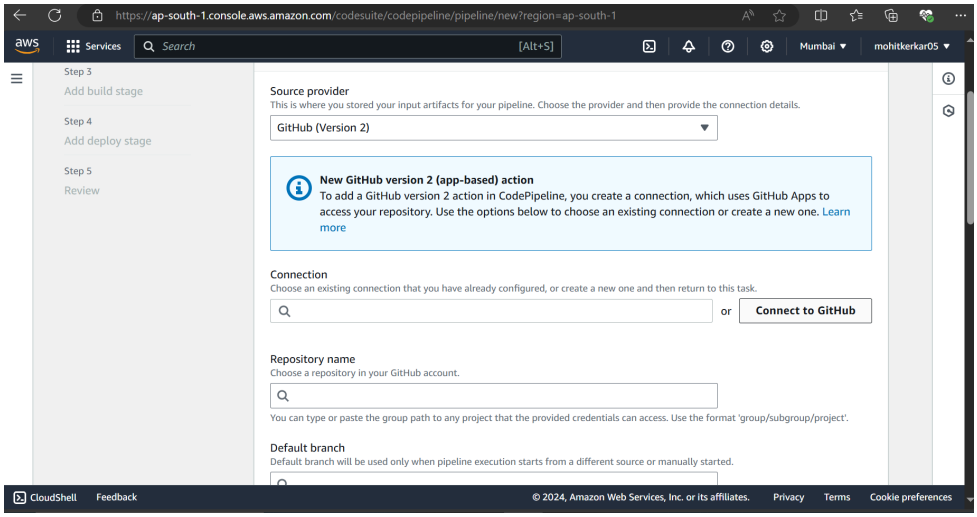
And click on next …



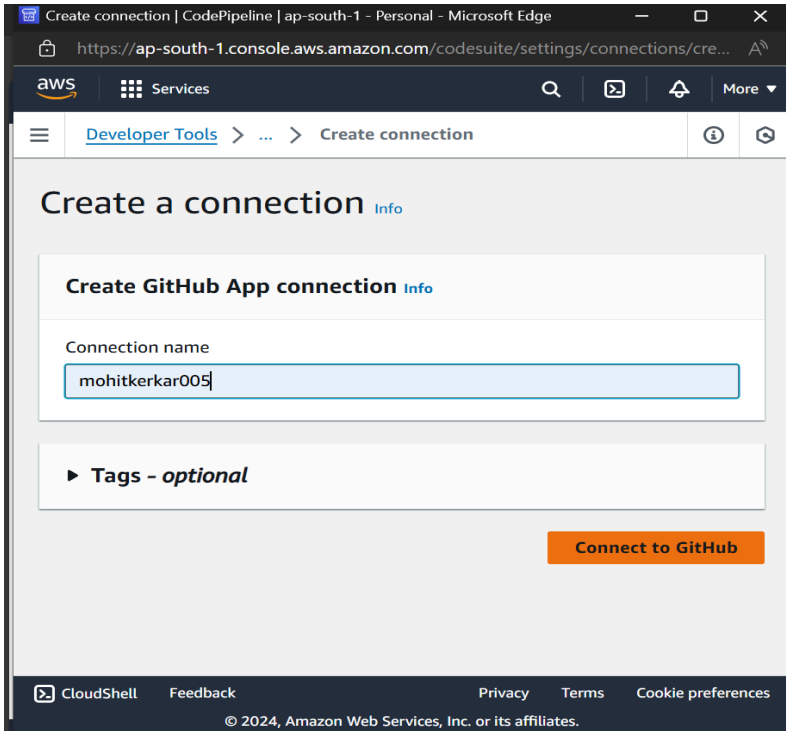
Name: Mohit Kerkar Div: D15C Roll No: 23

Step 4: Github connection

In this step, we are supposed to create a github connection and add our existing repository over here i.e the one we forked earlier

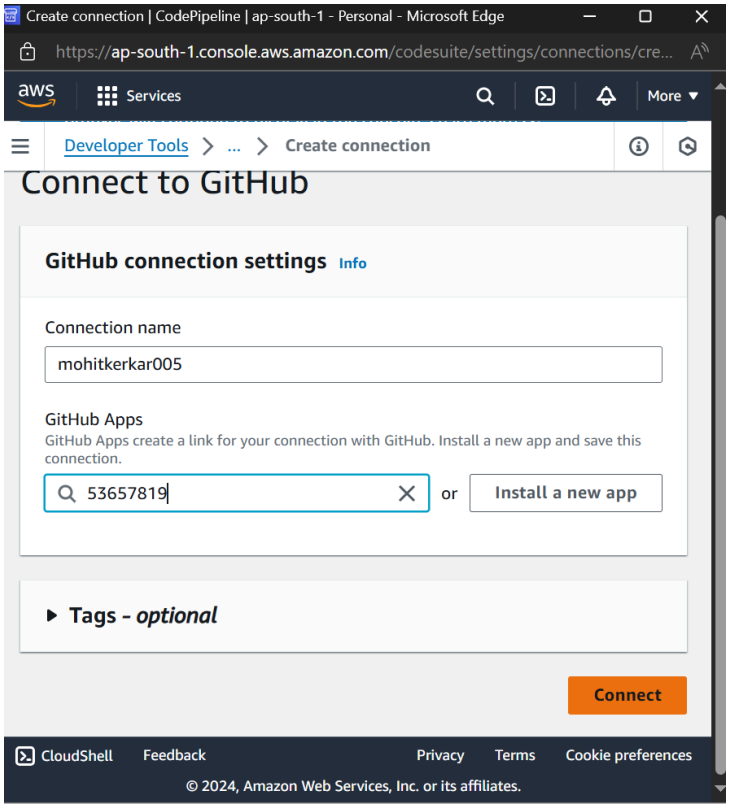


We are supposed to enter our github username so as to proceed towards making the connection

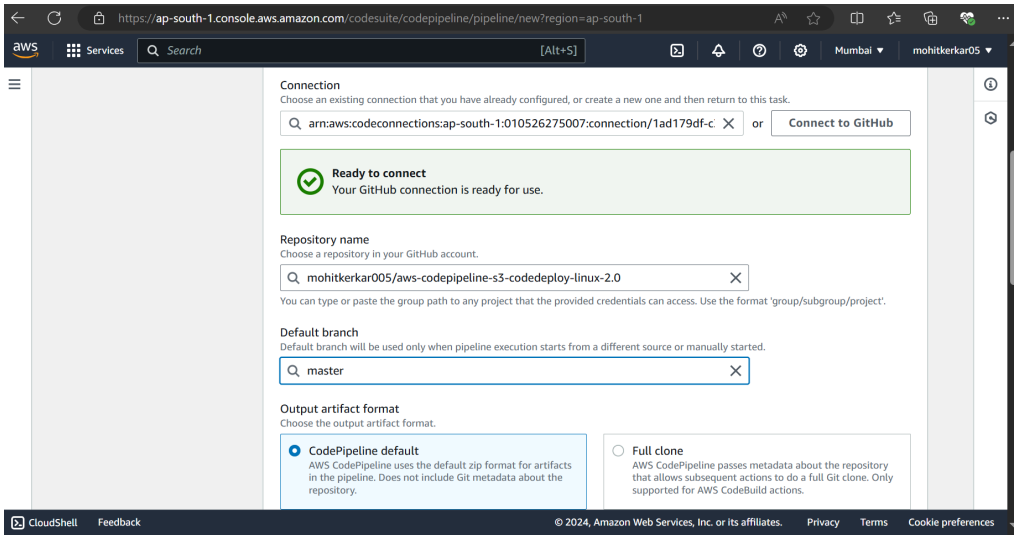


Now to finalize our connection, we are to install an application which connects AWS to our github account and repository.

Name: Mohit Kerkar Div: D15C Roll No: 23



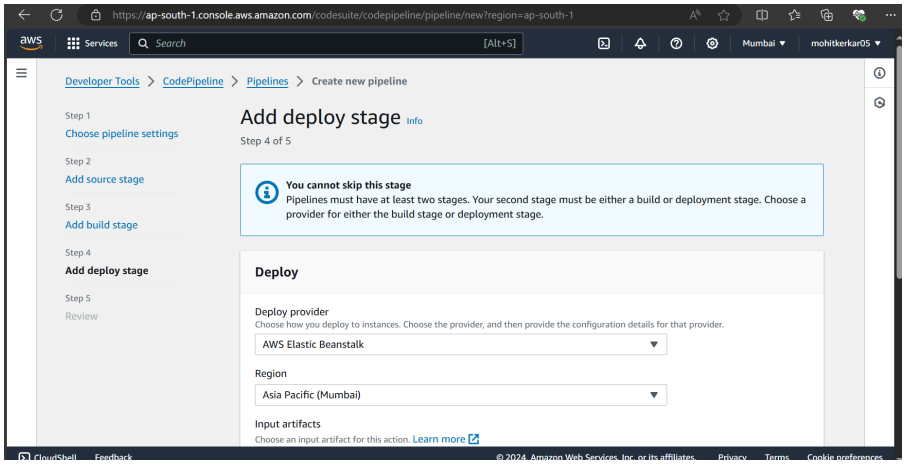
Post the establishment of the connection, this is the message that is displayed. We can further select the branch of our repository that we want to connect.

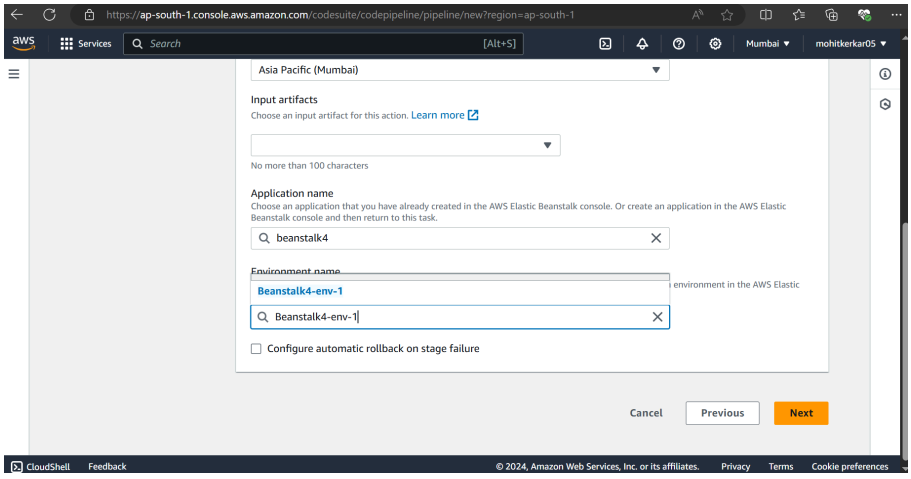


Step 5: Deployment stage:

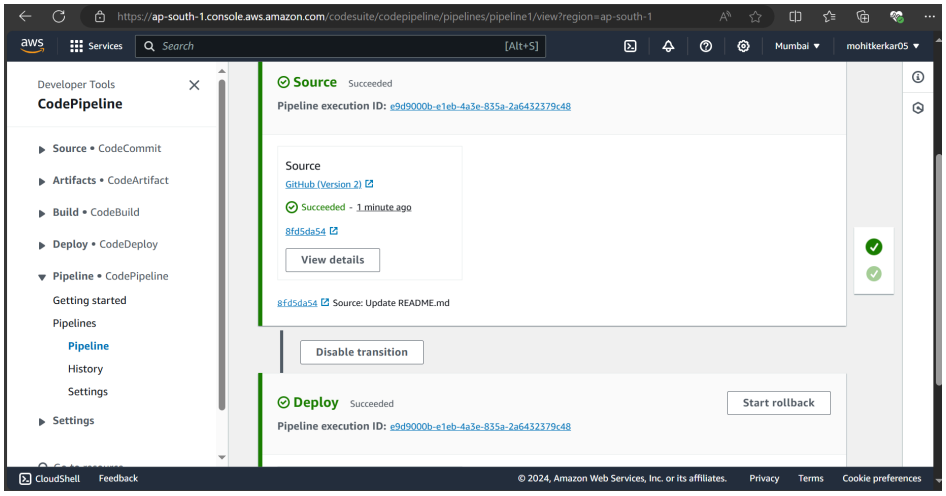
We are expected to skip the build stage and move towards the deployment step. In the deployment step we are supposed to choose the Elastic Beanstalk application and the environment that we created earlier and proceed with our pipeline creation

Name: Mohit Kerkar Div: D15C Roll No: 23

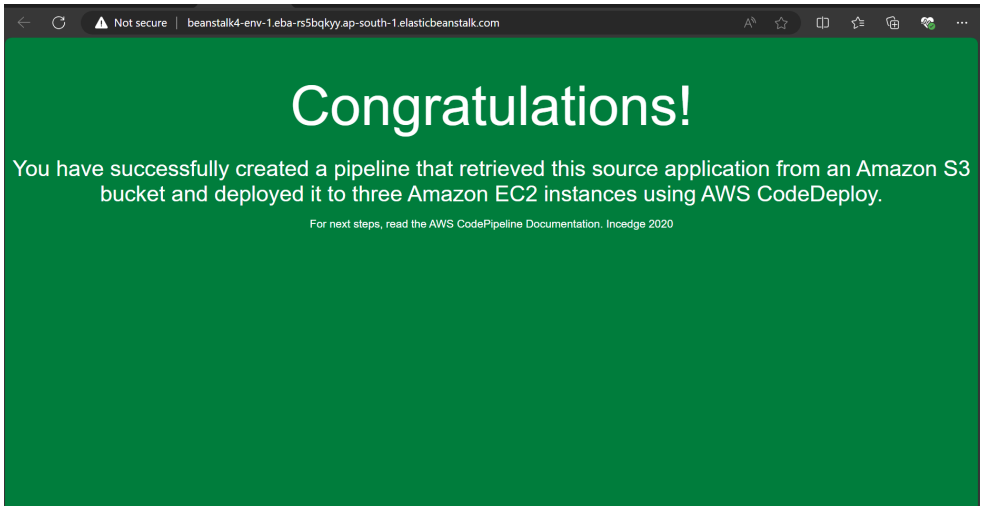




Step 6: Post deployment stage: When all the stages run successfully, this is what is displayed onto the screen. It shows us that our application and our environment have successfully been deployed using a dedicated pipeline created

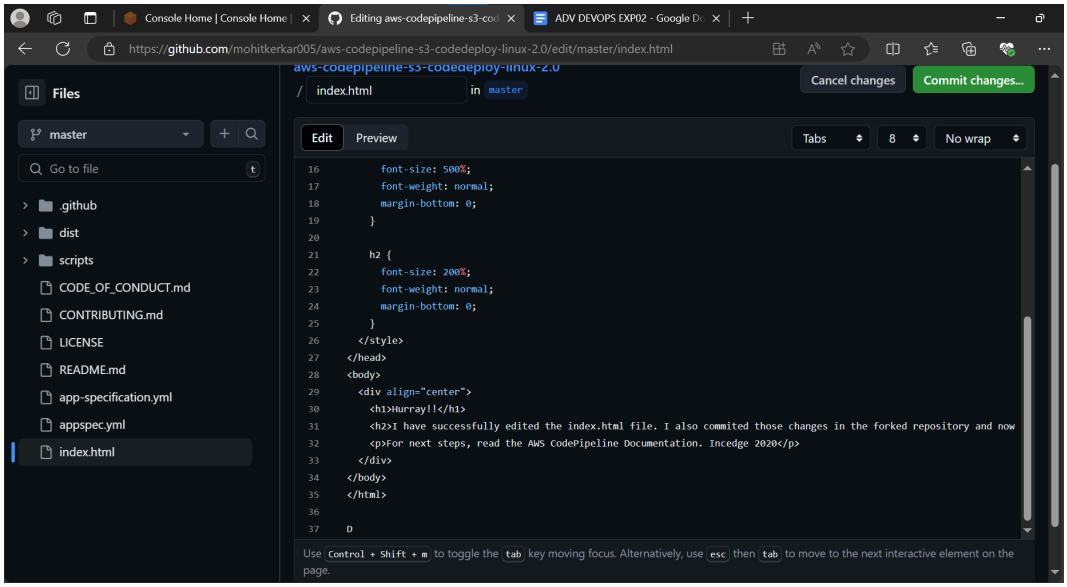


Name: Mohit Kerkar Div: D15C Roll No: 23



Step 7: Committing changes to your github code

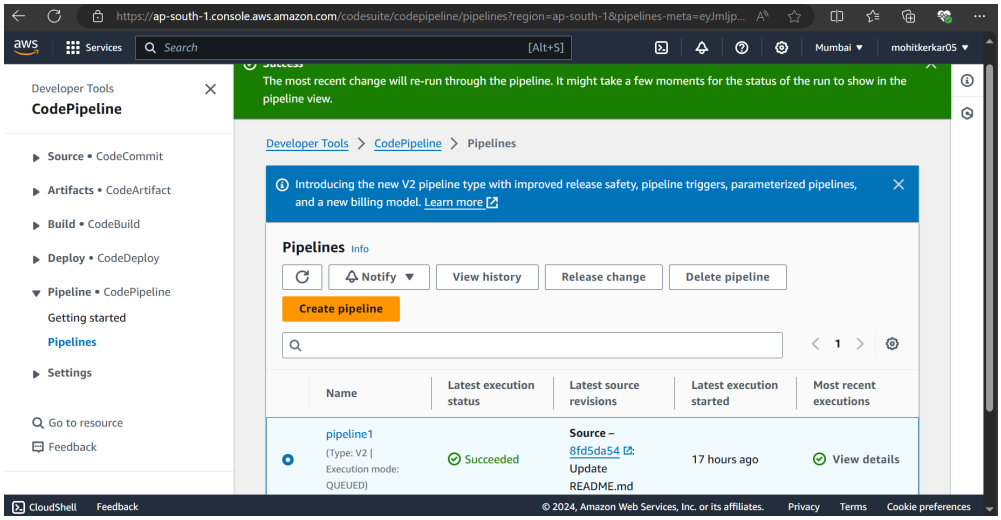
Now, we will go to our forked repository and make some changes to the index.html file. On making the desired changes, we are supposed to commit those changes on our forked repository. Write a good commit message so as to recognize it when it appears on the pipeline.



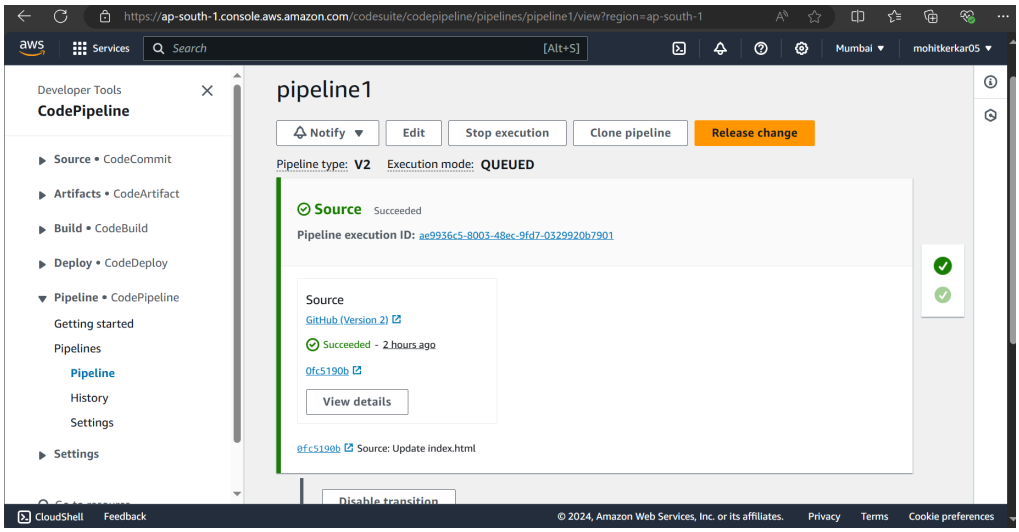
Step 8: Apply the newly made changes in index.html onto our pipeline

Come back to the Codepipeline section and select the pipeline through which we successfully created and deployed our application. Click on the release change option to apply the latest changes/commits from our github repository to our pipeline

Name: Mohit Kerkar Div: D15C Roll No: 23



Once the changes have been applied, we see the commit message that we wrote for the latest commit on our repository being reflected on our pipeline. Over here, it would be seen somewhere near the bottom of the image that is attached. “Update index.html” was the latest commit message in the github repository

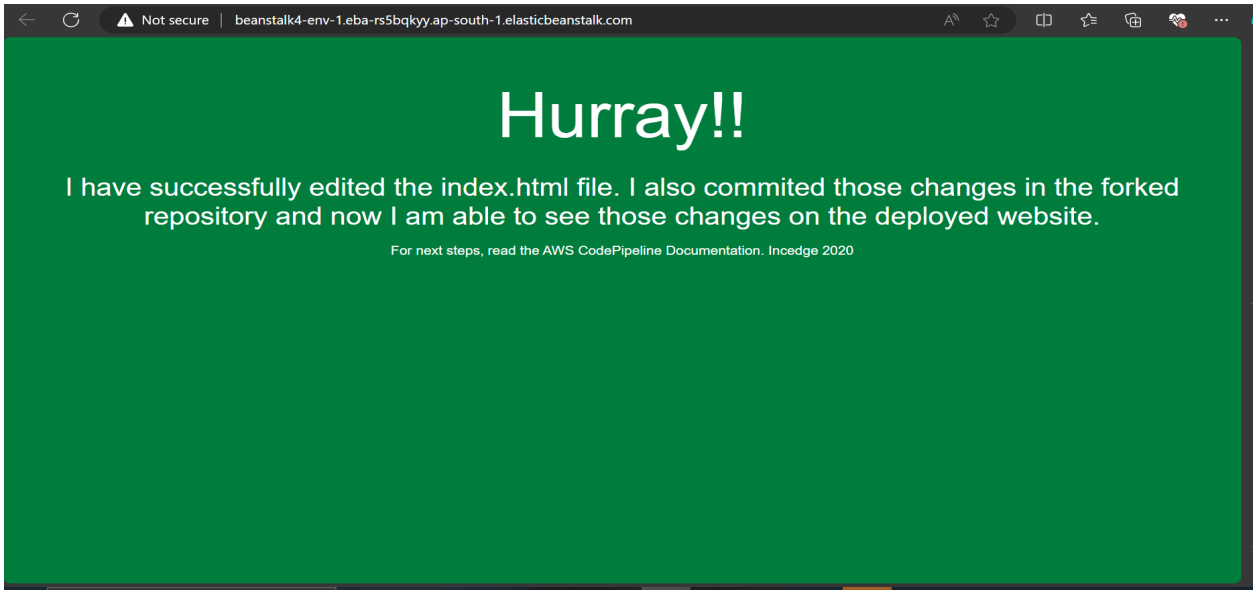


Step 9: Open the Domain of our Elastic Beanstalk environment

Now, we navigate back to our Elastic Beanstalk environment and open the environment domain of our deployed application

The text in this image is clearly distinguishable from the earlier website’s text meaning that the changes that we made to our code in index.html has successfully been applied to the website that we deployed

Name: Mohit Kerkar Div: D15C Roll No: 23



Conclusion: In conclusion, this experiment demonstrates how to build and deploy an application using AWS CodeBuild, CodePipeline, and CodeDeploy. We created an Elastic Beanstalk environment, configured it with the required IAM roles, and connected it to a GitHub repository through CodePipeline. By making changes to the application code in the GitHub repository, we were able to update the deployed application seamlessly. This process highlights the effectiveness of AWS services in automating deployment, streamlining code integration, and ensuring that updates are reflected quickly in production environments.

Name: Mohit Kerkar Div: D15C Roll No: 23

Adv DevOps Exp 03

Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up aKubernetes Cluster on Linux Machines/Cloud

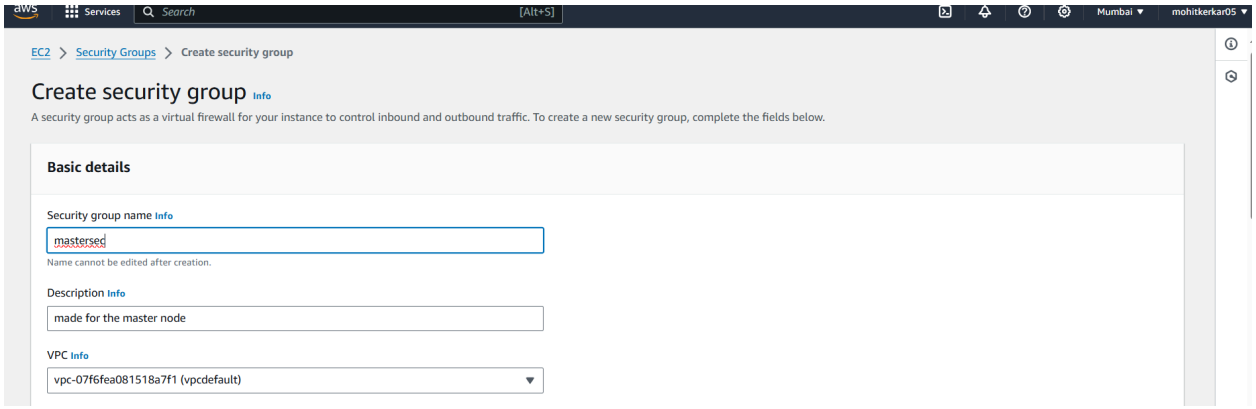
(I have performed this experiment on my personal AWS account)

Step 1: Create Key-pair, Security groups and required default VPCs

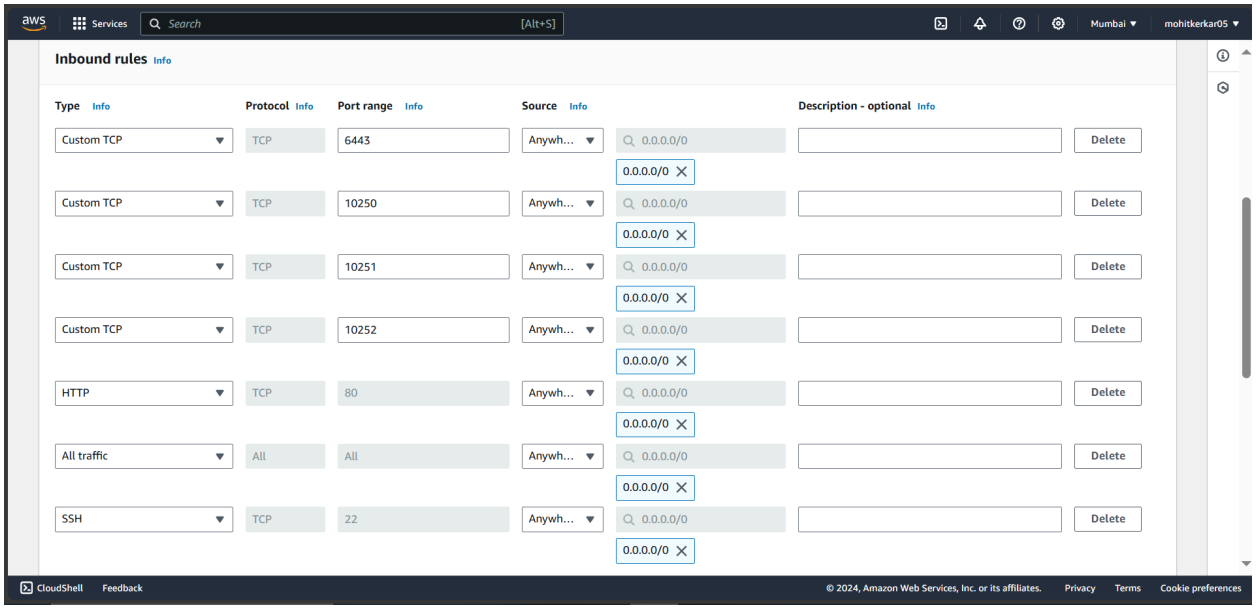
I started off by creating two separate security groups i.e one for the master instance (kubeadm to be initialized within it) and the other for the 2 worker instances. For creating any EC2 security groups, we require a VPC (Virtual private cloud) and a subnet along with it .. so that we can work with or allow inbound and outbound traffic and communication. Make sure you have a default VPC and a subnet already created which can be used for this experiment.

Also, make sure that you have a key pair installed of the type RSA with .pem extension on your local machine. Save it at a place which is accessible and where you can work from your terminal.

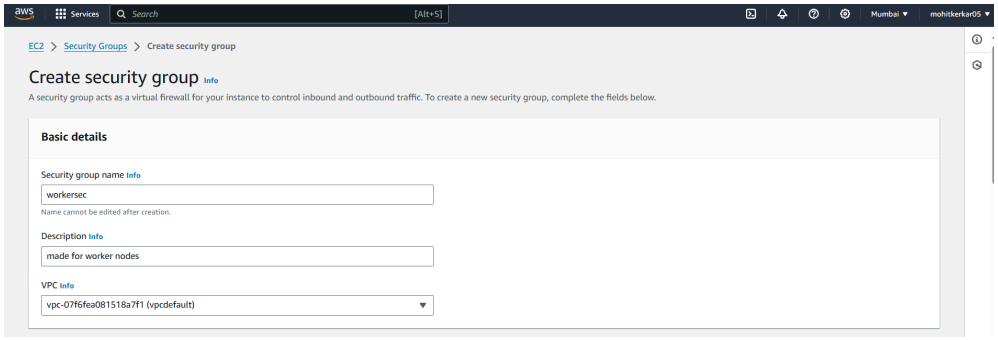
Here, i created my first security group



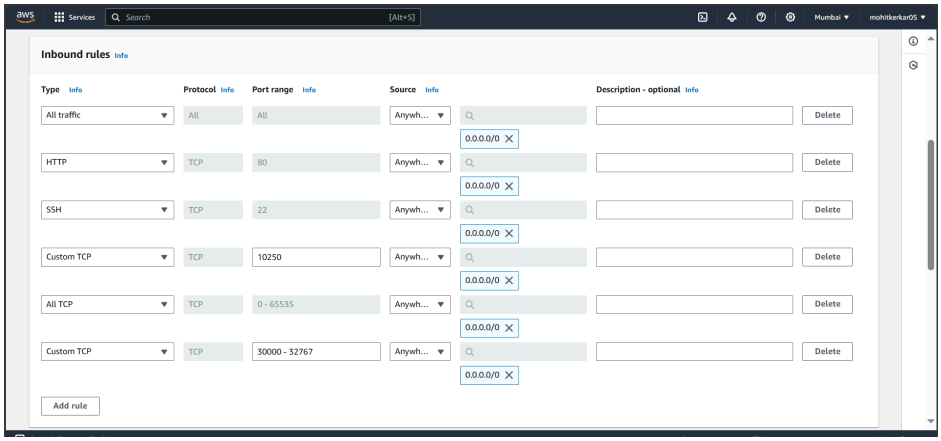
I modified the inbound rules in the following fashion for the master node



Then, i created a security group for the worker nodes



Edited the inbound rules for the worker nodes in the following fashion

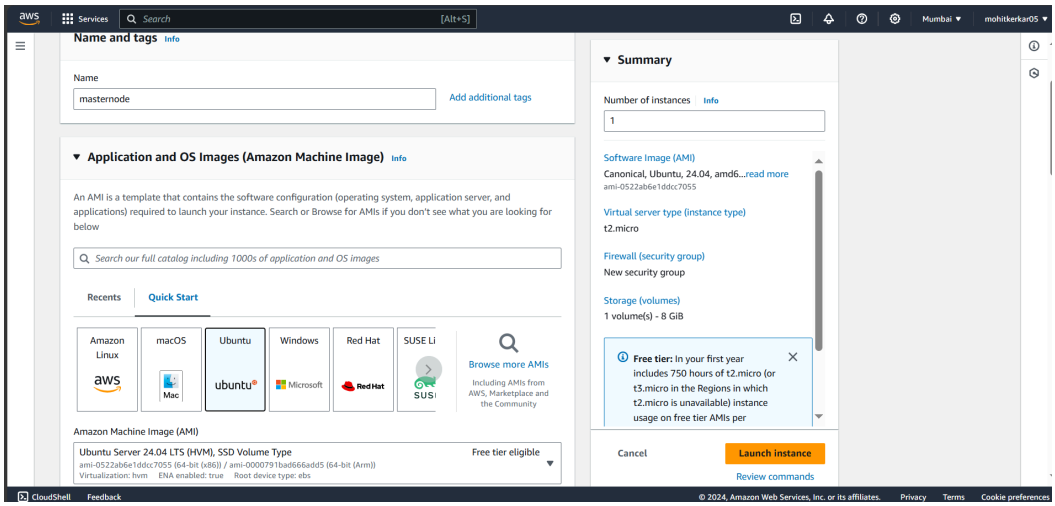


Step 2: Create 3 EC2 instances i.e one master and other 2 workers

Now, in order to work with the instances, navigate to EC2 instances section and launch a few instances. Launch one instance and name it as masternode and the other two as worker.

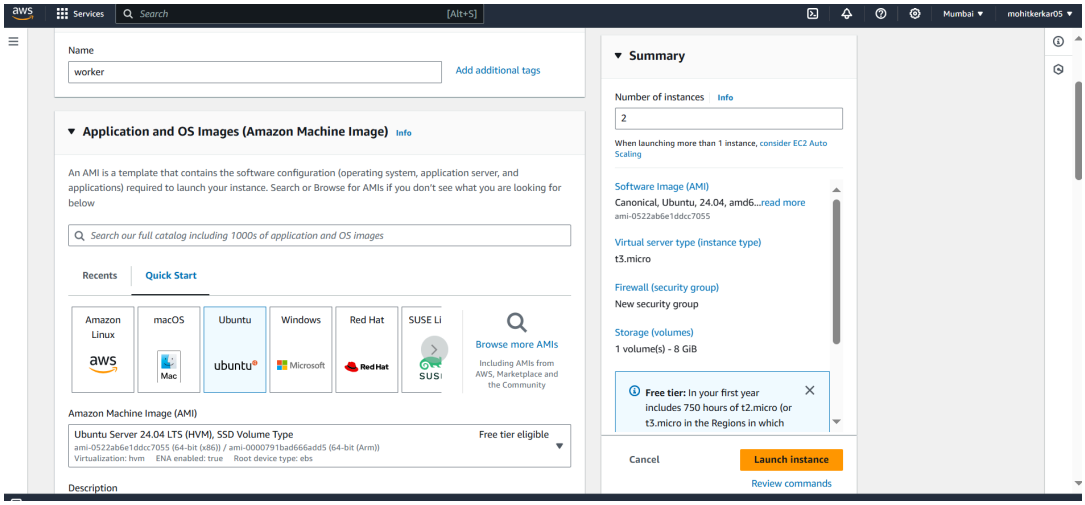
I selected **AMI as ubuntu** among the list of OS that we shown available in the free tier eligibility list

Following is the master instance node:

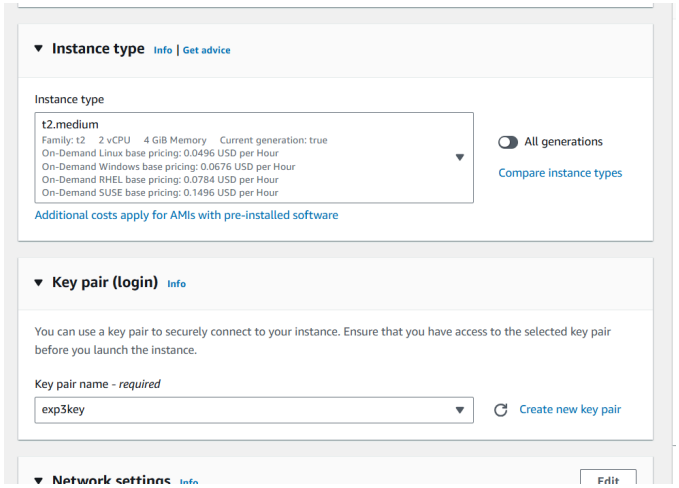


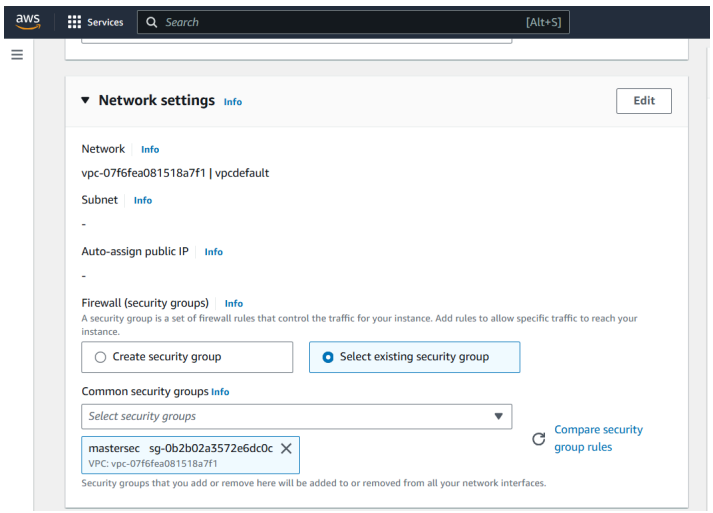
Name: Mohit Kerkar Div: D15C Roll No: 23

Following is the worker instance node:



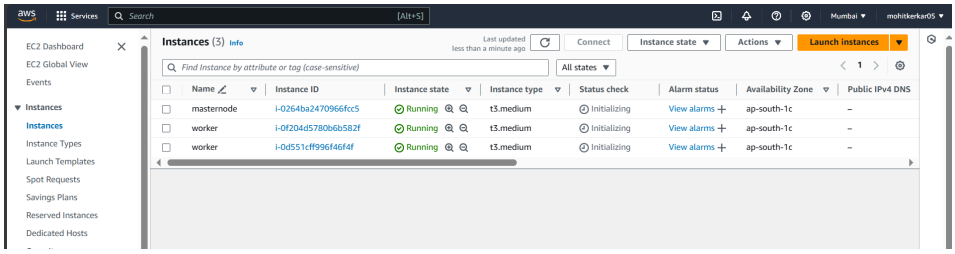
IMPORTANT: Select t2.medium as the instance type for all 3 instances, as kubernetes requires at least 2 CPUs to work with and sufficient amount of other resources as well



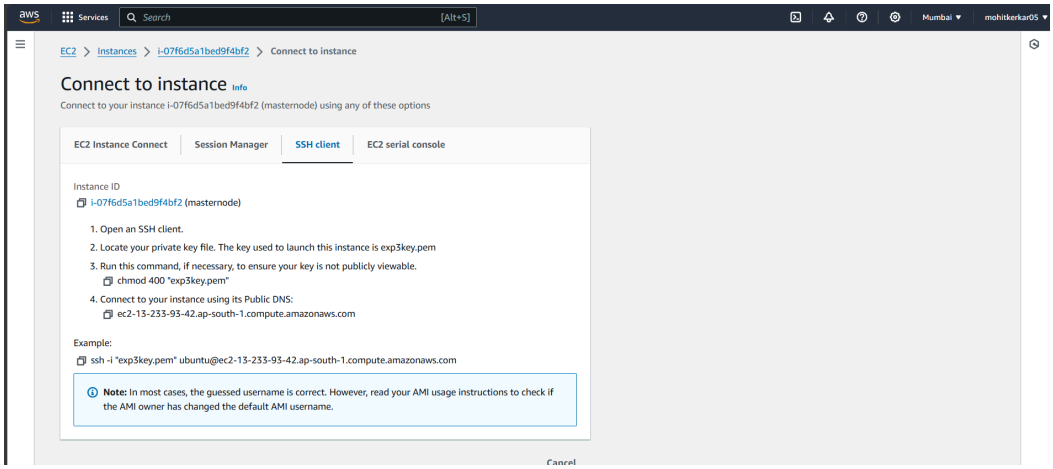


Name: Mohit Kerkar Div: D15C Roll No: 23

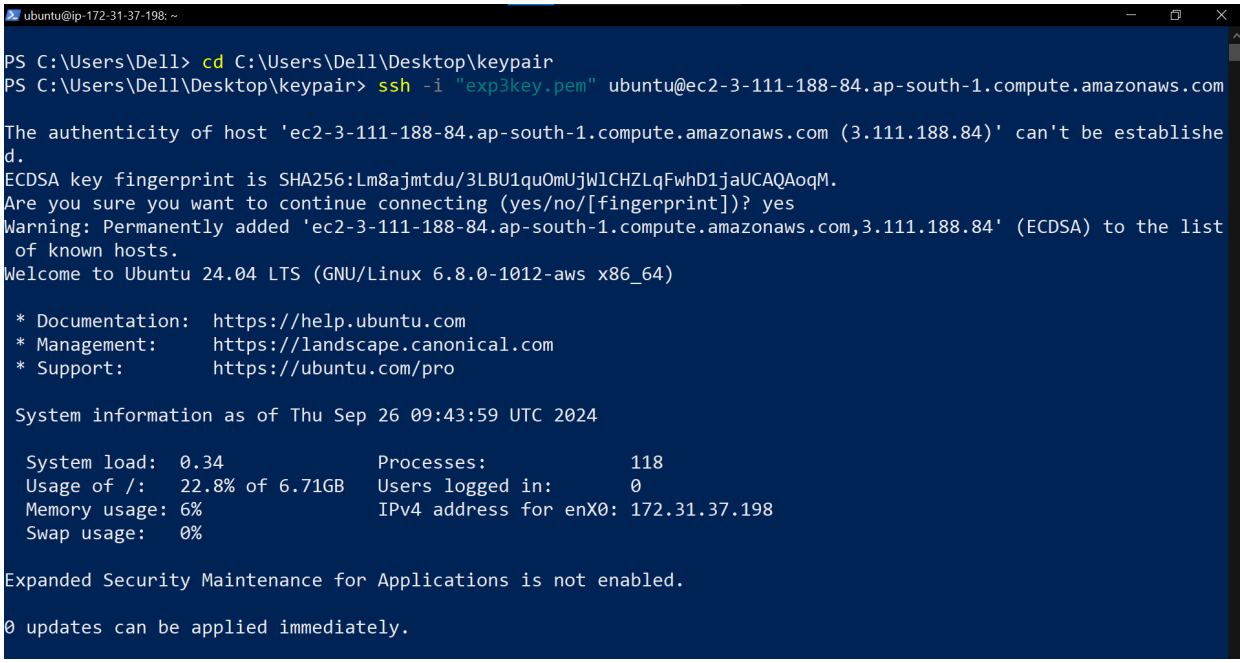
After launching all the 3 instances, select one instance and press ‘Connect’



Navigate to SSH client section and copy the command thats listed for connecting to the node remotely
ssh -i "exp3key.pem" ubuntu@ec2-13-233-93-42.ap-south-1.compute.amazonaws.com…for the master node
Similarly, do this for all nodes



Now, open 3 separate terminals and run change directories to the folder which contains the key we created earlier
Run their respective SSH commands (one in each terminal)
This helps us log onto those instances individually and remotely and work on them separately



```
PS C:\Windows\system32> cd C:\Users\Dell\Desktop\keypair
PS C:\Users\Dell\Desktop\keypair> ssh -i "exp3key.pem" ubuntu@ec2-13-234-38-84.ap-south-1.compute.amazonaws.com

The authenticity of host 'ec2-13-234-38-84.ap-south-1.compute.amazonaws.com (13.234.38.84)' can't be establishe
d.
ECDSA key fingerprint is SHA256:P0+5r+bZ6OGpc0sL2A0+kkZYBNHMjfucgne/ZBHPMHQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-13-234-38-84.ap-south-1.compute.amazonaws.com,13.234.38.84' (ECDSA) to the list
 of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Sep 26 09:46:24 UTC 2024

  System load:  0.08              Processes:             118
  Usage of /:   22.8% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.47.161
  Swap usage:   0%
```

```
PS C:\Windows\system32> cd C:\Users\Dell\Desktop\keypair
PS C:\Users\Dell\Desktop\keypair> ssh -i "exp3key.pem" ubuntu@ec2-65-0-74-51.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-65-0-74-51.ap-south-1.compute.amazonaws.com (65.0.74.51)' can't be established.
ECDSA key fingerprint is SHA256:RImwOJI2RUM5s3vV2bYLL681AJE41Easupz4bVTCjFw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-65-0-74-51.ap-south-1.compute.amazonaws.com,65.0.74.51' (ECDSA) to the list of
known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Sep 26 09:47:05 UTC 2024

  System load:  0.13              Processes:             119
  Usage of /:   22.8% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.33.106
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.
```

Step 3: Docker installation:

Run the following command: These commands are used to install Docker on an Ubuntu system by adding Docker's official GPG key and configuring the Docker repository.

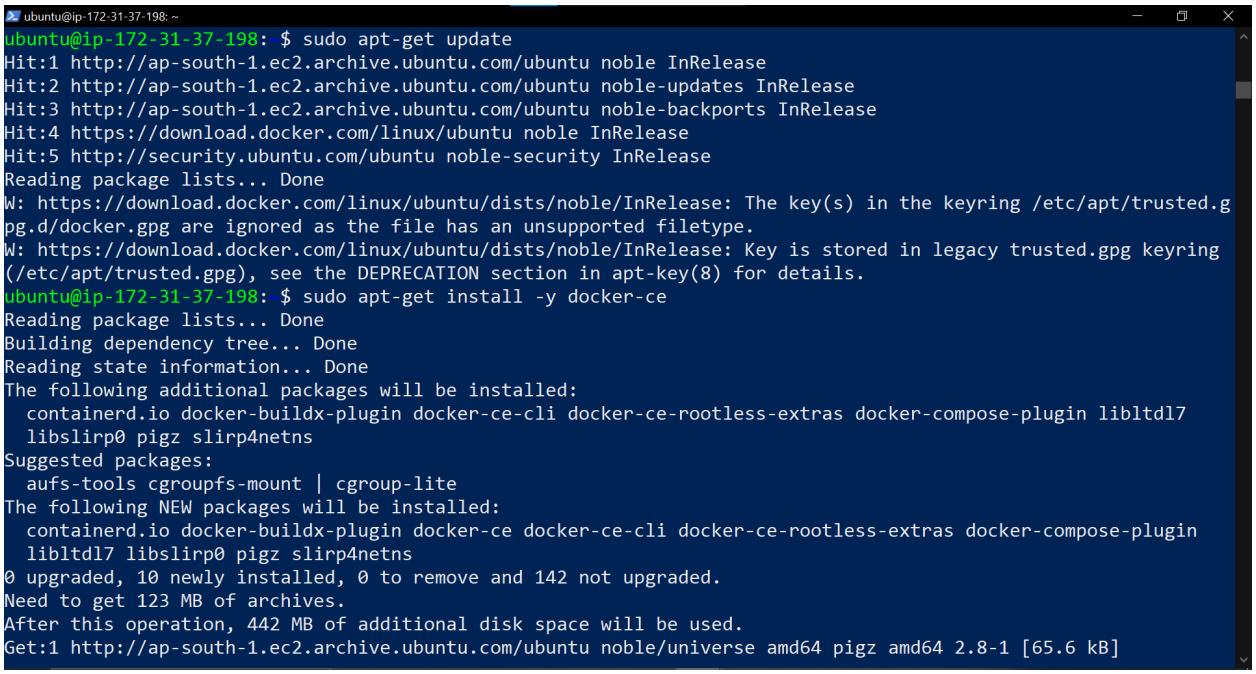
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

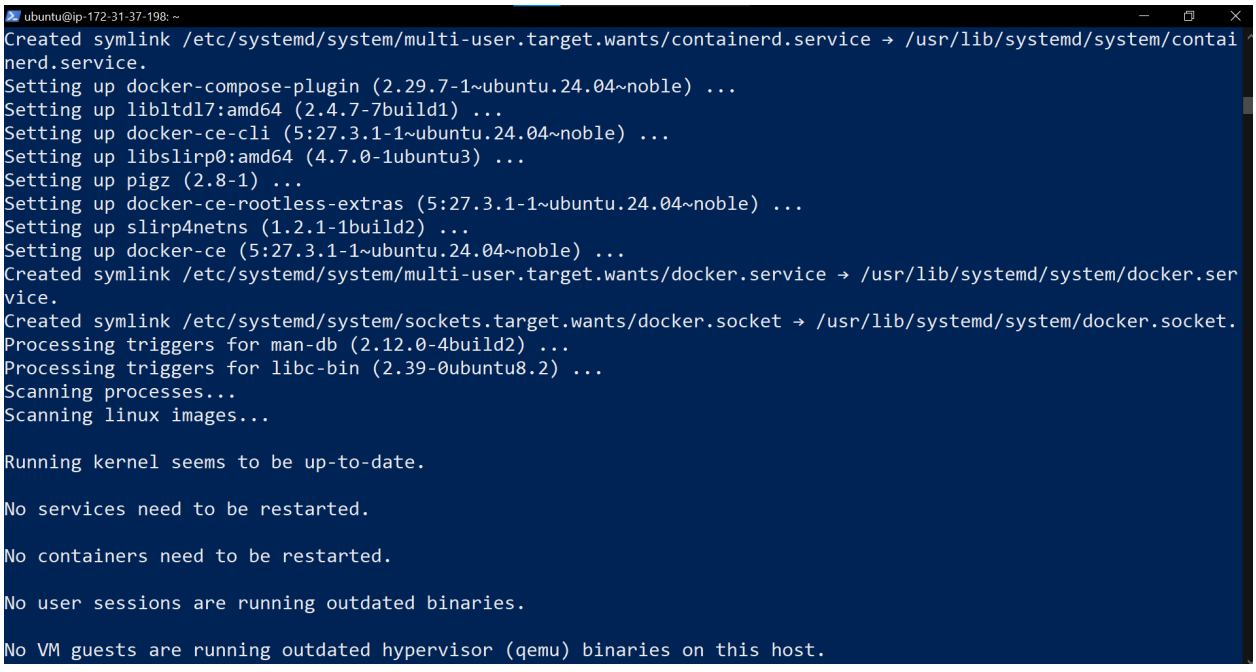
```
ubuntu@ip-172-31-37-198: ~
ubuntu@ip-172-31-37-198: $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.
gpg.d/docker.gpg > /dev/null
ubuntu@ip-172-31-37-198: $ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
> $(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-nob
le.list
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:8 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [15.3 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:11 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:12 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:13 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:14 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
```

Name: Mohit Kerkar Div: D15C Roll No: 23

Run the following commands to refresh your local package list to ensure the latest packages are available and install Docker Community Edition on your system without prompting for confirmation.

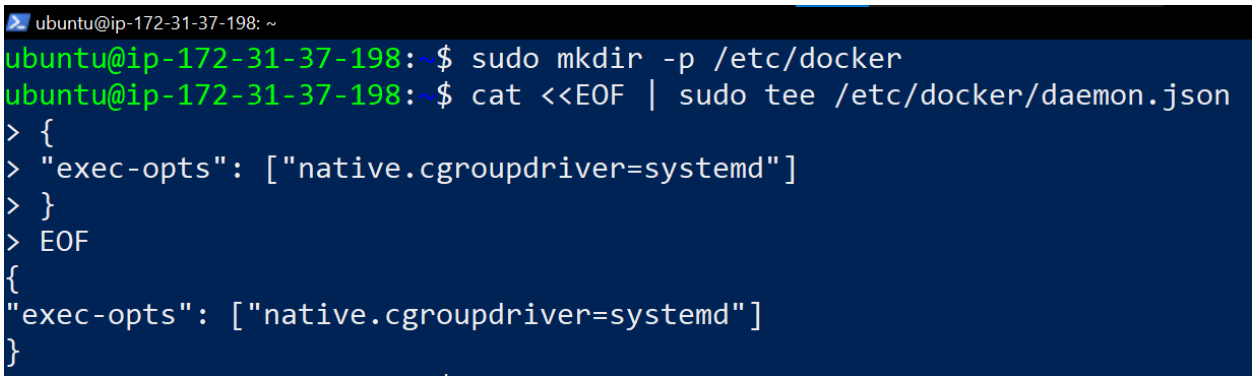
sudo apt-get update
sudo apt-get install -y docker-ce





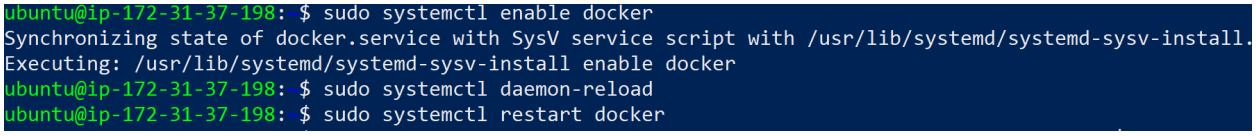
Run the following commands to create the Docker configuration directory and write a configuration file for Docker to use the systemd cgroup driver.

sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF



Run the following commands to ensure Docker starts automatically on system boot, reload systemd to apply new configurations, restart Docker to apply the new configuration.

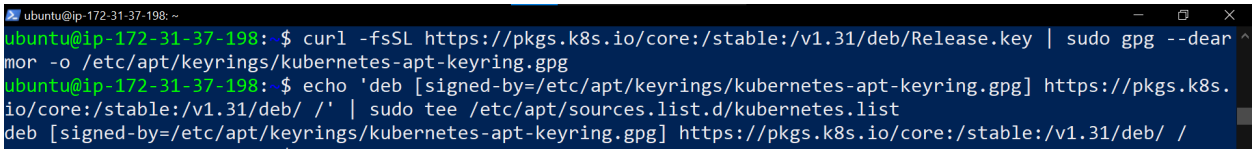
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker



Step 4: Kubernetes Installation:

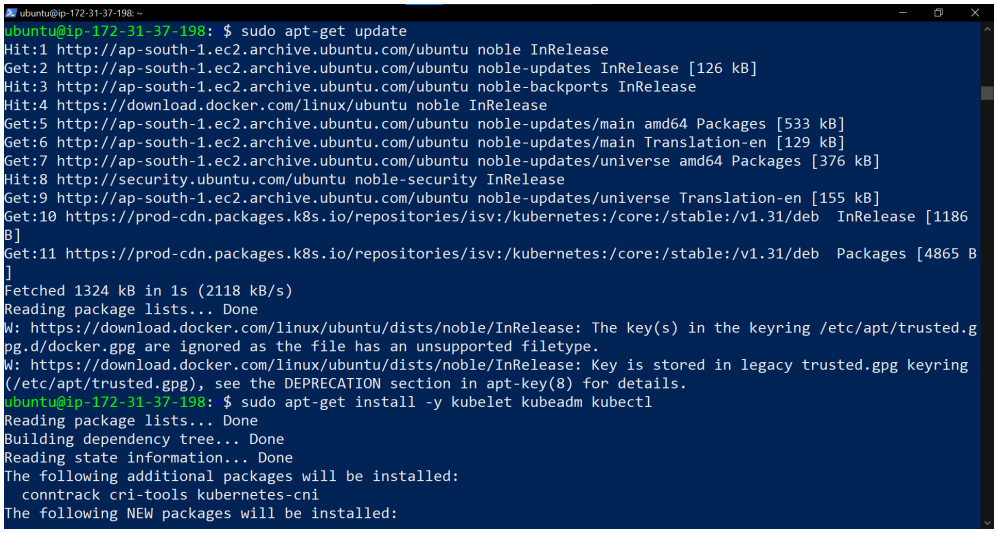
Run the following commands to add the Kubernetes package repository to your Ubuntu system in order to install Kubernetes components like kubectl, kubelet, and kubeadm.

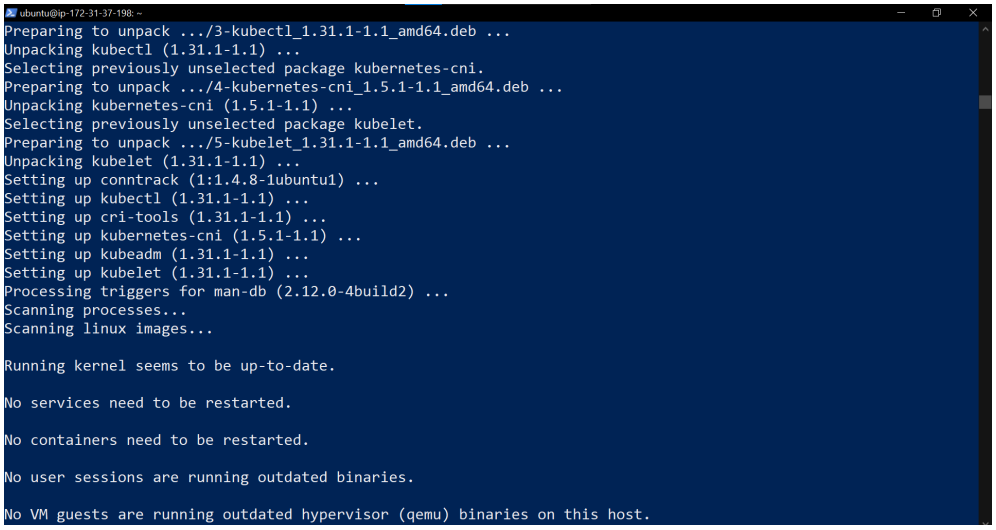
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

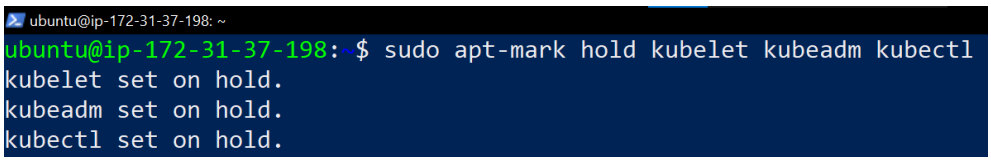


Next, run these commands to refresh the local package index to include the newly added Kubernetes repository, install the main Kubernetes components (kubelet, kubeadm, kubectl), prevent the installed Kubernetes components from being automatically updated, ensuring cluster stability until manual updates are performed.

sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl

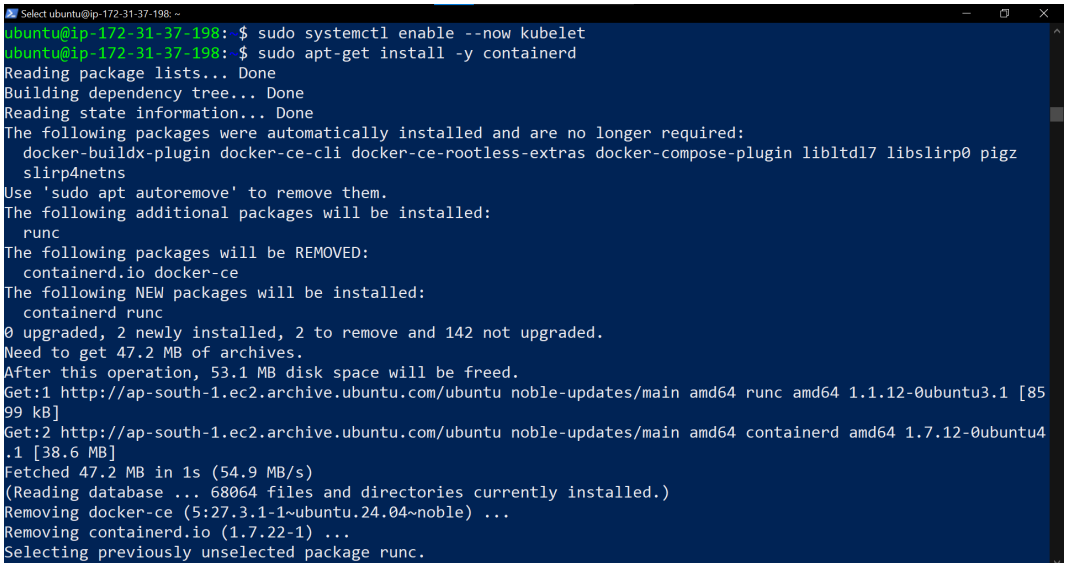




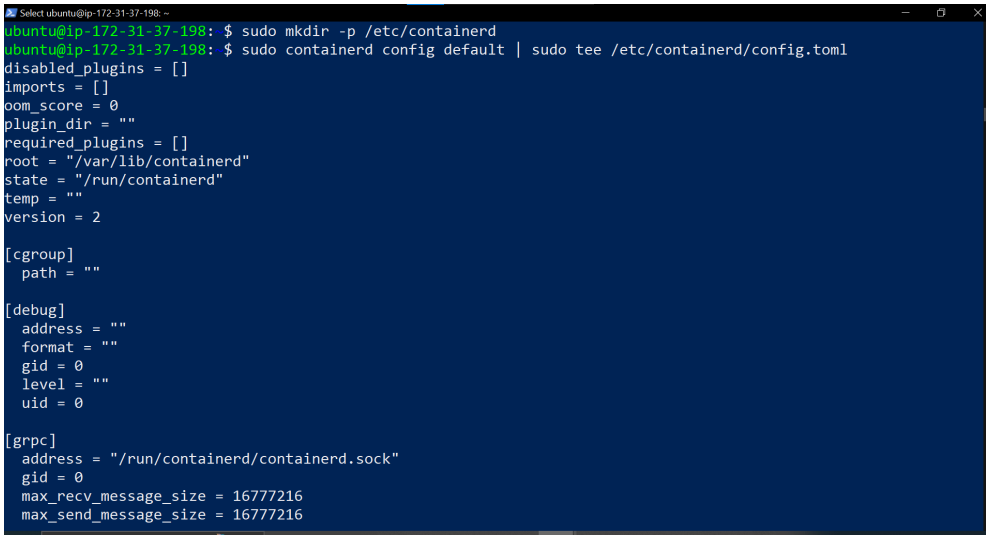


The command **sudo systemctl enable --now kubelet** enables the **kubelet** service, ensuring it starts automatically at boot and immediately starts running without needing a system reboot. The **sudo apt-get install -y containerd** command installs **containerd**, a container runtime that Kubernetes uses to manage and run containers.

sudo systemctl enable --now kubelet
sudo apt-get install -y containerd



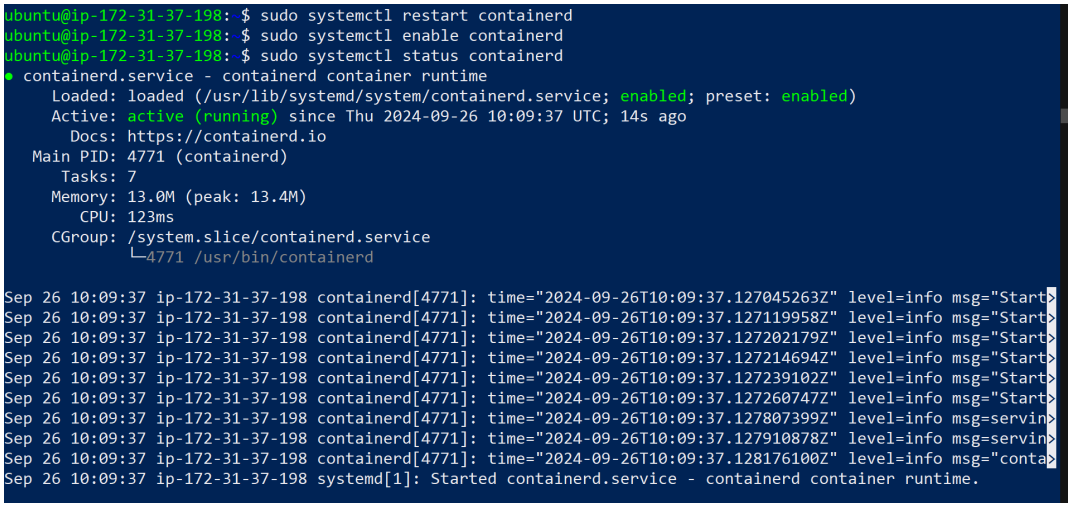
The command `sudo mkdir -p /etc/containerd` creates the directory for containerd configuration files if it doesn't already exist. The next command, `sudo containerd config default | sudo tee /etc/containerd/config.toml`, generates a default configuration for containerd and saves it as `config.toml` in that directory. Together, these commands set up the necessary directory and create a default configuration file for containerd.
sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml



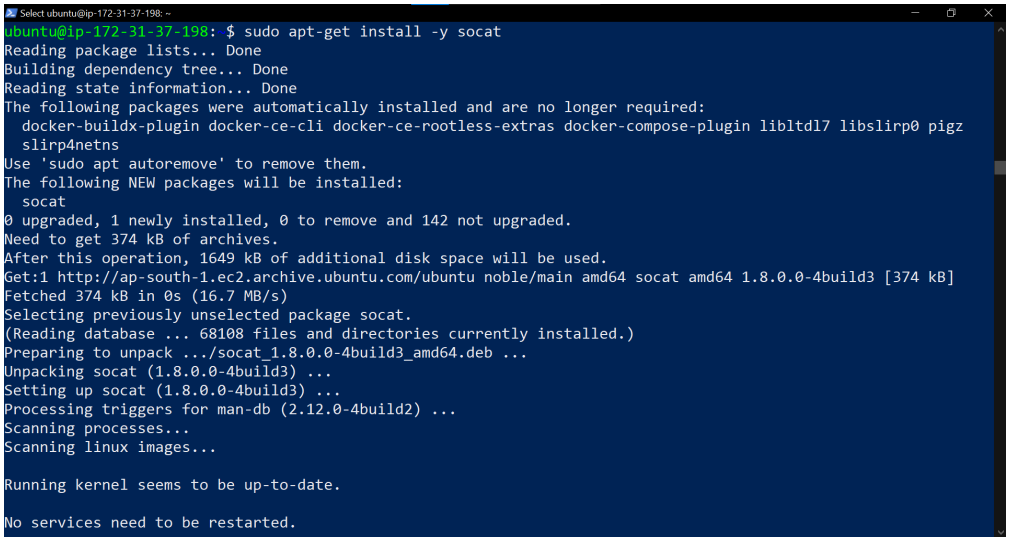
Name: Mohit Kerkar Div: D15C Roll No: 23

The command `sudo systemctl restart containerd` restarts the `containerd` service to apply any changes. `sudo systemctl enable containerd` sets it to start automatically at boot, while `sudo systemctl status containerd` checks its current status, showing whether it's active and any errors. Together, these commands manage the operation and health of the container runtime.

sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd



Socat installation:
sudo apt-get install -y socat

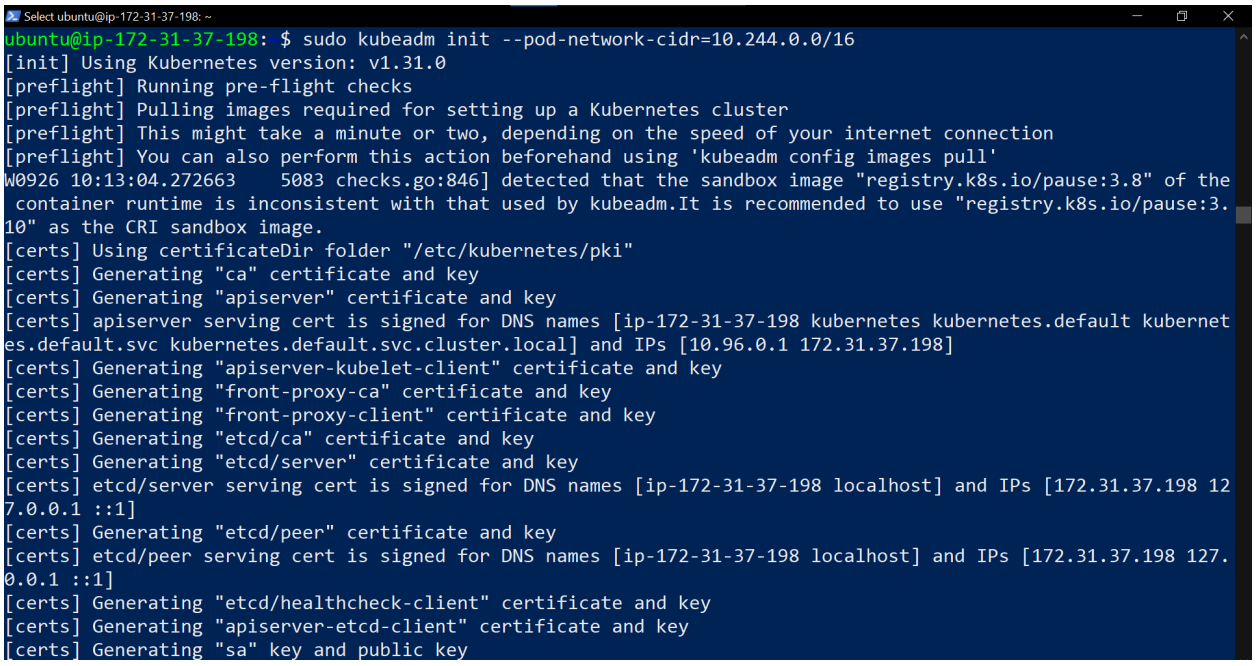


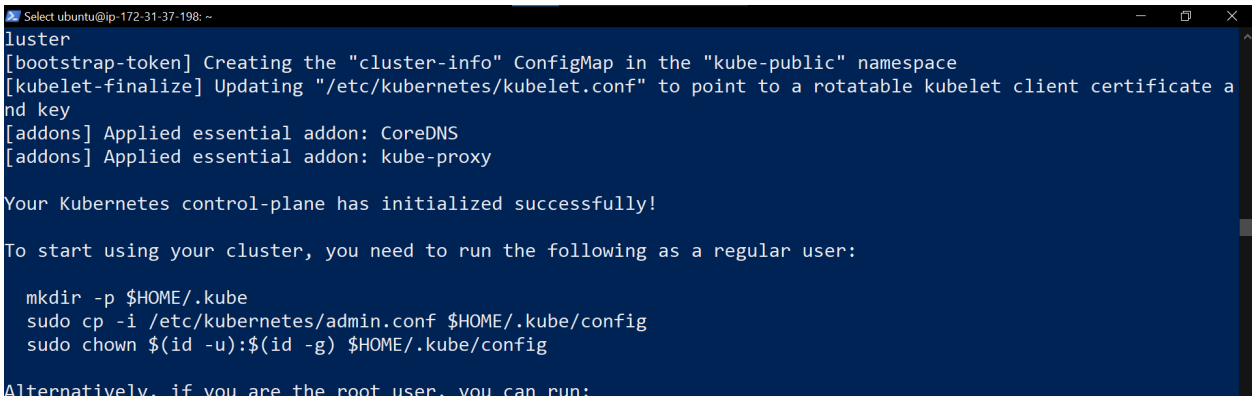
Step 5: Kubernetes Cluster

Run this command only on the master instance

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

It initializes a Kubernetes cluster with kubeadm and sets up the control plane (master node).



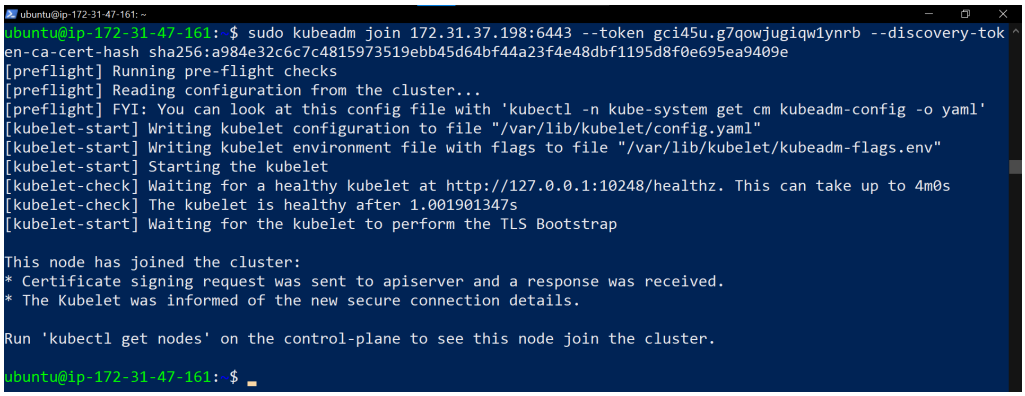


Run this command on master and also copy and save the Join command from above.
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

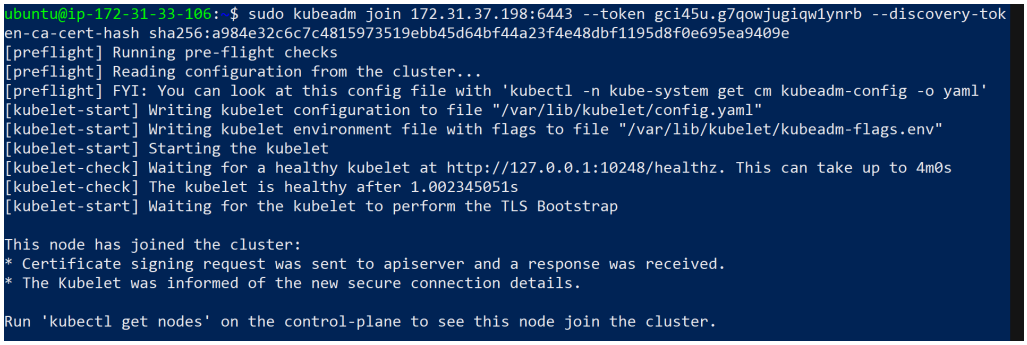
```
ubuntu@ip-172-31-37-198:~$ mkdir -p $HOME/.kube
ubuntu@ip-172-31-37-198:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
ubuntu@ip-172-31-37-198:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-37-198:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE   VERSION
ip-172-31-37-198   NotReady   control-plane   12m   v1.31.1
```

Connect master and worker nodes by running this command on the worker logged in terminals:
I ran the following command to do this,
kubeadm join 172.31.37.198:6443 --token gci45u.g7qowjugiqw1ynrb \ --discovery-token-ca-cert-hash sha256:a984e32c6c7c4815973519ebb45d64bf44a23f4e48dbf1195d8f0e695ea9409e

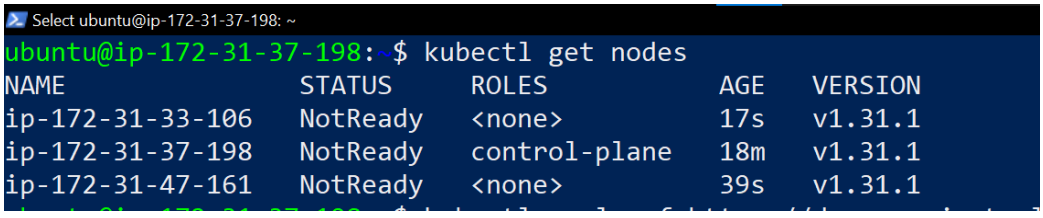
On node 1:



On node 2:

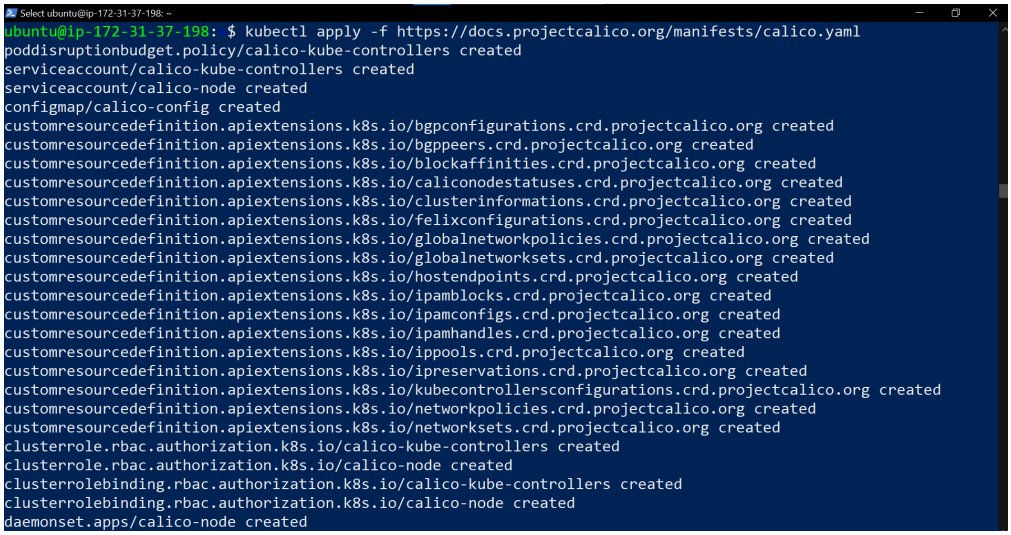


Run kubectl get nodes on the master instance to confirm the joining of the worker nodes

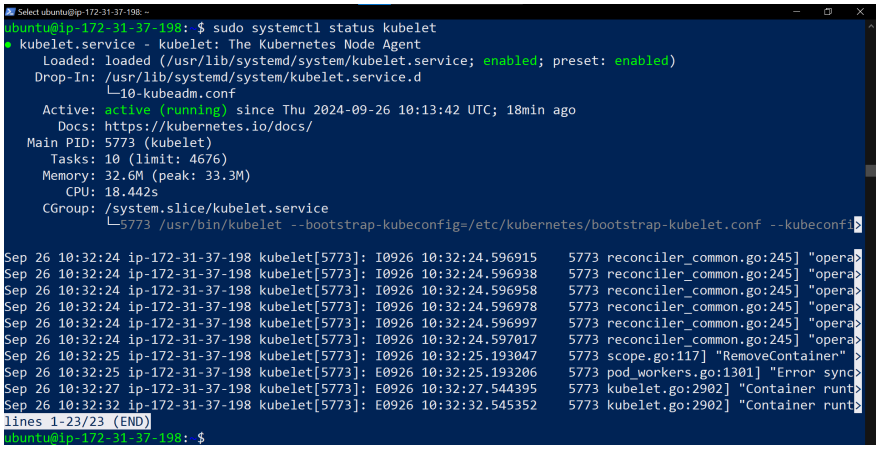


Since Status is NotReady we have to add a network plugin. And also we have to give the name to the nodes.
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

The command kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml deploys Calico, a networking and network security solution for Kubernetes, by applying the configuration specified in the provided YAML file from the Calico documentation. This sets up the necessary resources to enable networking capabilities within the Kubernetes cluster.



sudo systemctl status kubelet



kubectl get nodes -o wide helps us get to know that the Status is ready.

Name: Mohit Kerkar Div: D15C Roll No: 23

```
ubuntu@ip-172-31-37-198:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE           KE
RNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-33-106   Ready    <none>          66s   v1.31.1   172.31.33.106   <none>        Ubuntu 24.04 LTS   6.
8.0-1012-aws   containerd://1.7.12
ip-172-31-37-198   Ready    control-plane   19m   v1.31.1   172.31.37.198   <none>        Ubuntu 24.04 LTS   6.
8.0-1012-aws   containerd://1.7.12
ip-172-31-47-161   Ready    <none>          88s   v1.31.1   172.31.47.161   <none>        Ubuntu 24.04 LTS   6.
8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-37-198:~$ kubectl label node ip-172-31-28-117 kubernetes.io/role=Node1
Error from server (NotFound): nodes "ip-172-31-28-117" not found
ubuntu@ip-172-31-37-198:~$ kubectl label node ip-172-31-47-161 kubernetes.io/role=Node1
node/ip-172-31-47-161 labeled
ubuntu@ip-172-31-37-198:~$ kubectl label node ip-172-31-33-106 kubernetes.io/role=Node2
node/ip-172-31-33-106 labeled
ubuntu@ip-172-31-37-198:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE
KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-33-106   Ready    Node2           3m26s   v1.31.1   172.31.33.106   <none>        Ubuntu 24.04 LTS
6.8.0-1012-aws   containerd://1.7.12
ip-172-31-37-198   Ready    control-plane   21m     v1.31.1   172.31.37.198   <none>        Ubuntu 24.04 LTS
6.8.0-1012-aws   containerd://1.7.12
ip-172-31-47-161   Ready    Node1           3m48s   v1.31.1   172.31.47.161   <none>        Ubuntu 24.04 LTS
6.8.0-1012-aws   containerd://1.7.12
```

Or else run kubectl get nodes command

```
ubuntu@ip-172-31-37-198:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-33-106    Ready    Node2           3m42s   v1.31.1
ip-172-31-37-198    Ready    control-plane   21m     v1.31.1
ip-172-31-47-161    Ready    Node1           4m4s    v1.31.1
ubuntu@ip-172-31-37-198:~$
```

Conclusion: In this experiment, we successfully set up a Kubernetes cluster on AWS by creating the necessary infrastructure and configuring security groups. We installed Docker and Kubernetes components, initialized the master node, and connected the worker nodes. By deploying Calico for networking, we enabled effective communication within the cluster. This hands-on experience enhanced our understanding of Kubernetes architecture and equipped us with practical skills for managing containerized applications in a cloud environment.

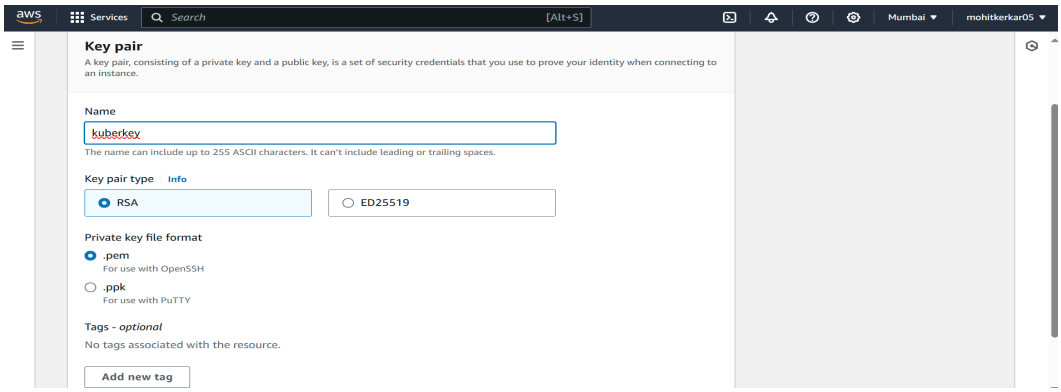
Name: Mohit Kerkar Div: D15C Roll No: 23

Adv DevOps Exp 04

Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application. (I performed this Experiment on my personal AWS account)

Step 1: Creation of EC2 Instance, its Key-pair, remote login using SSH

Firstly, create a key pair with a desired name and type 'RSA' with .pem extension. (.pem is useful with OpenSSH)

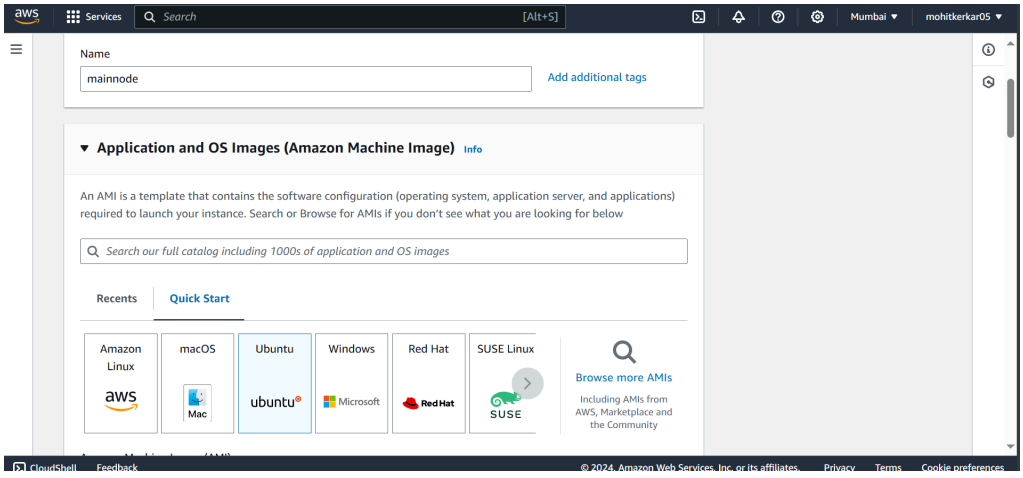


After creating the key, it automatically gets downloaded onto your machine. We are to create a new folder and place our within there. After this, we are to access our key and establish our connection with our instance by changing the directory to our key's directory

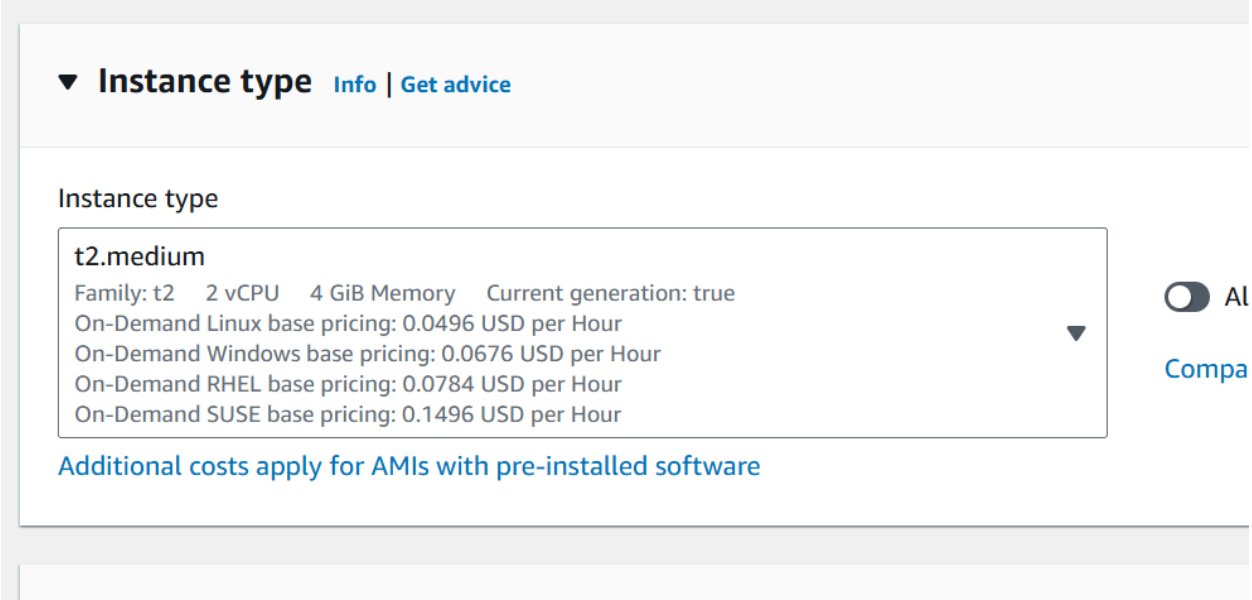
Now, proceed to the creation of the EC2 instance. I selected Amazon machine image as **Ubuntu** and instance type as **t2.medium**, just for the purpose of convenience and also for successful execution of the experiment.

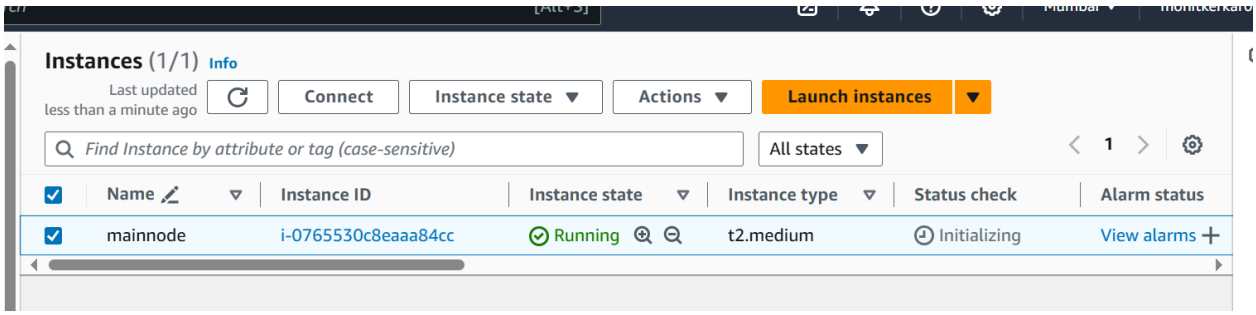
The execution of the experiment is determined by the fact that how many resources is the instance allowed to access or use. The Free Tier eligible instance type **t2.micro avails the instance with only one** CPU, whereas with t2.medium we get 2 CPU and much more execution space.

Be mindful that along with all this, **t2.medium does not have free tier eligibility**. So, use the instances and the resources that you allow them to access resourcefully and economically. Shut/terminate them as and when work related to them is over

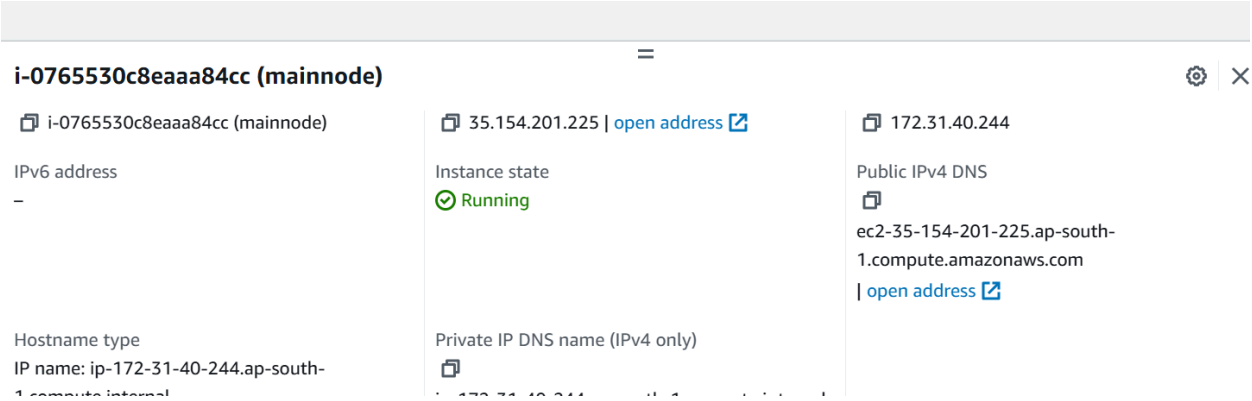


Name: Mohit Kerkar Div: D15C Roll No: 23



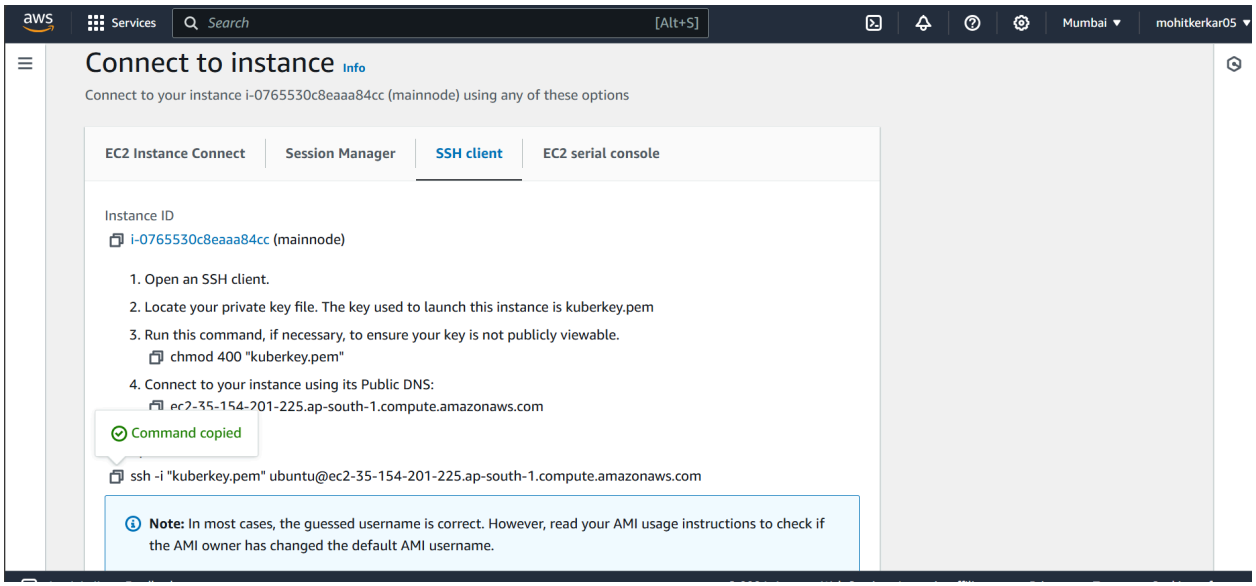


Here are my instance details. From here on, after remotely logging in from my local terminal, you would see my IPv4 address in my terminal (Powershell)

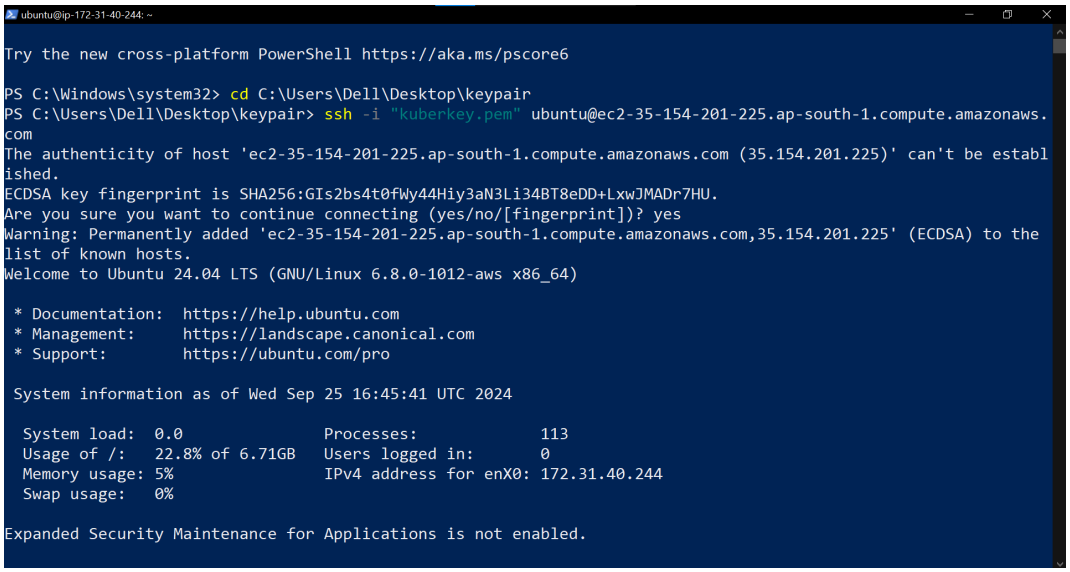


Name: Mohit Kerkar Div: D15C Roll No: 23

Select the instance and press connect. Navigate to the SSH client section and copy the SSH command provided to us. We will require this for remotely logging into our instance from our local terminal.



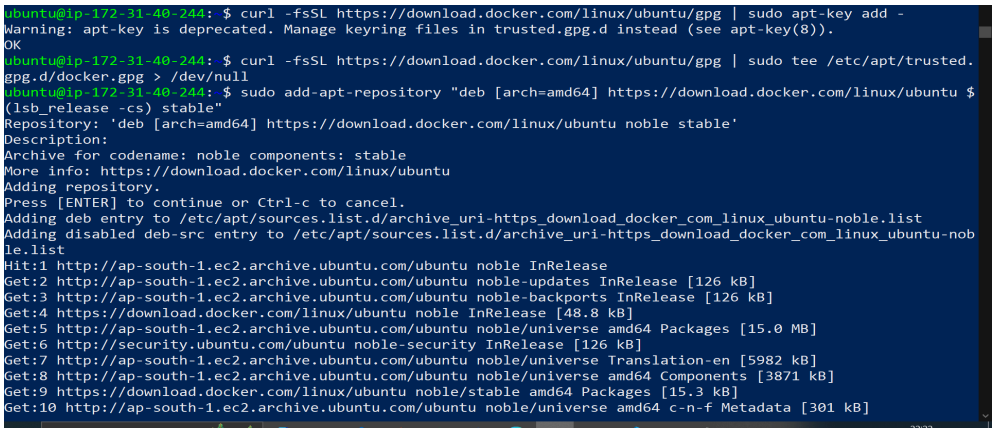
As one can see, I changed my directory to the folder in which i had stored my key (.pem) and ran the ssh command so as to remotely login onto my instance and perform the necessary tasks from my terminal itself



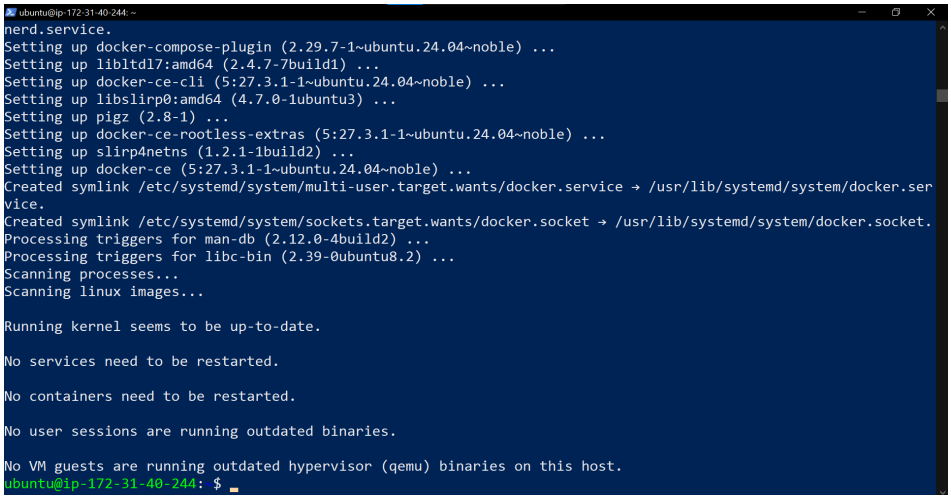
Step 2: Docker installation

Run the below mentioned commands in order to install docker on your instance

Adding key: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
Adding key without apt-key: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
Add docker repository: sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"



Run these commands to update the package list and install Docker:
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io



Next, run this command….
sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF

sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker

```
ubuntu@ip-172-31-40-244:~$ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-40-244:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-40-244:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-40-244:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-40-244:~$ sudo systemctl restart docker
```

Step 3: Kubernetes Installation

Run these commands to install kubernetes:
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-40-244:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dear
mor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
ubuntu@ip-172-31-40-244:~$ echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.
io/core:/stable:/v1.31/deb/ /" | Out-File -FilePath "C:\path\to\your\directory\kubernetes.list" -Encoding utf8
Out-File: command not found
ubuntu@ip-172-31-40-244:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
> https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-40-244:~$
```

After this step, run the following commands,
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl

Over here, while running sudo apt-get update if you encounter an error like this

```
ubuntu@ip-172-31-40-244:~$ sudo apt-get update
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
```

Then run the following command and ensure that the file contains the following texts:
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /

```
ubuntu@ip-172-31-40-244: ~
  GNU nano 7.2                    /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

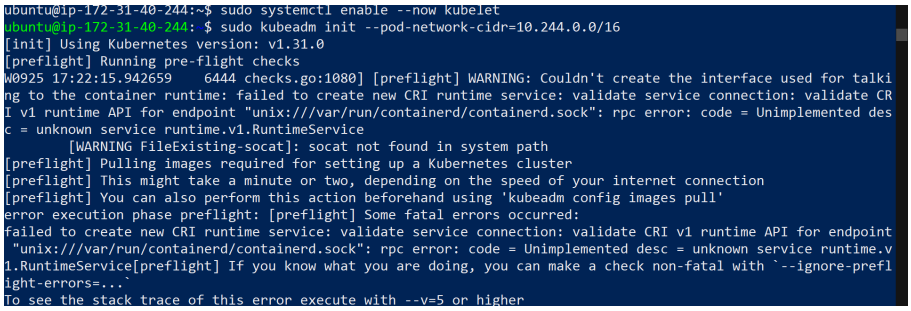
Be sure to save this file and then proceed with these commands

sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl

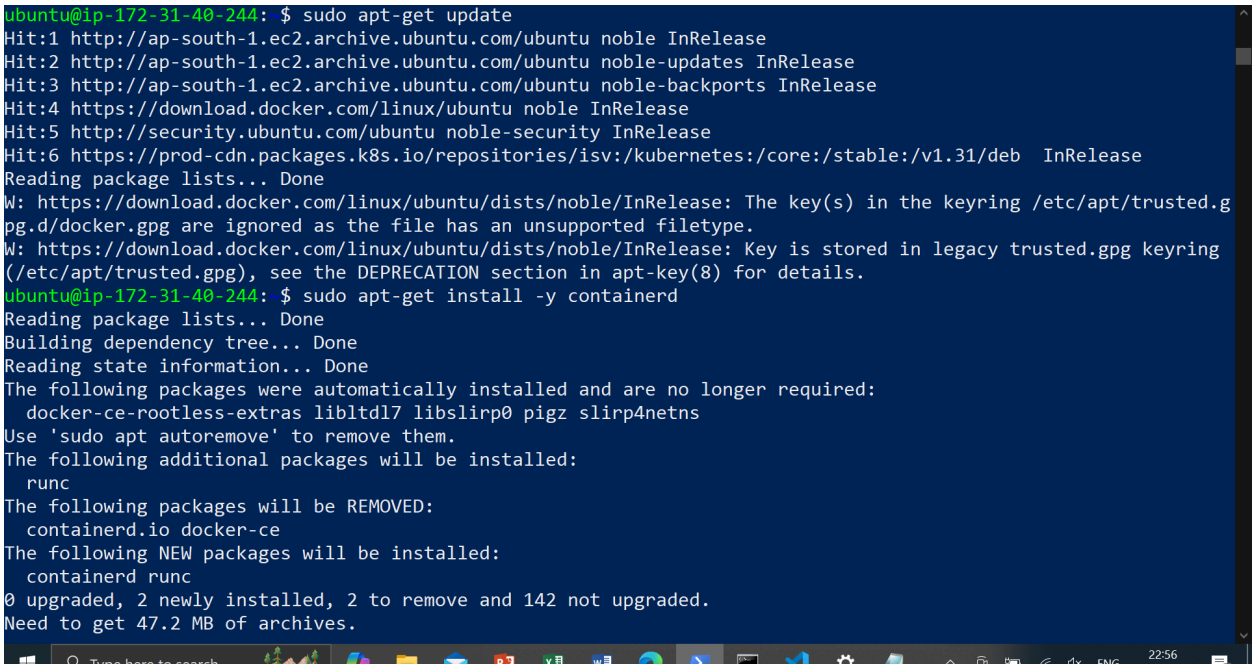
```
ubuntu@ip-172-31-40-244: ~
ubuntu@ip-172-31-40-244:~$ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B
]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 132 kB in 1s (199 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.g
pg.d/docker.gpg are ignored as the file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring
(/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
ubuntu@ip-172-31-40-244:~$ sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 142 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 k
```

```
Setting up conntrack (1:1.4.8-1ubuntu1) ...
Setting up kubectl (1.31.1-1.1) ...
Setting up cri-tools (1.31.1-1.1) ...
Setting up kubernetes-cni (1.5.1-1.1) ...
Setting up kubeadm (1.31.1-1.1) ...
Setting up kubelet (1.31.1-1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-40-244:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-40-244:~$
```

sudo systemctl enable --now kubelet
sudo kubeadm init --pod-network-cidr=10.244.0.0/16



Since running that command ran us into an error, we are expected to install containerd first
Run this command: sudo apt-get install -y containerd
Then run,
sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml



Post this step, run these commands one by one
sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd

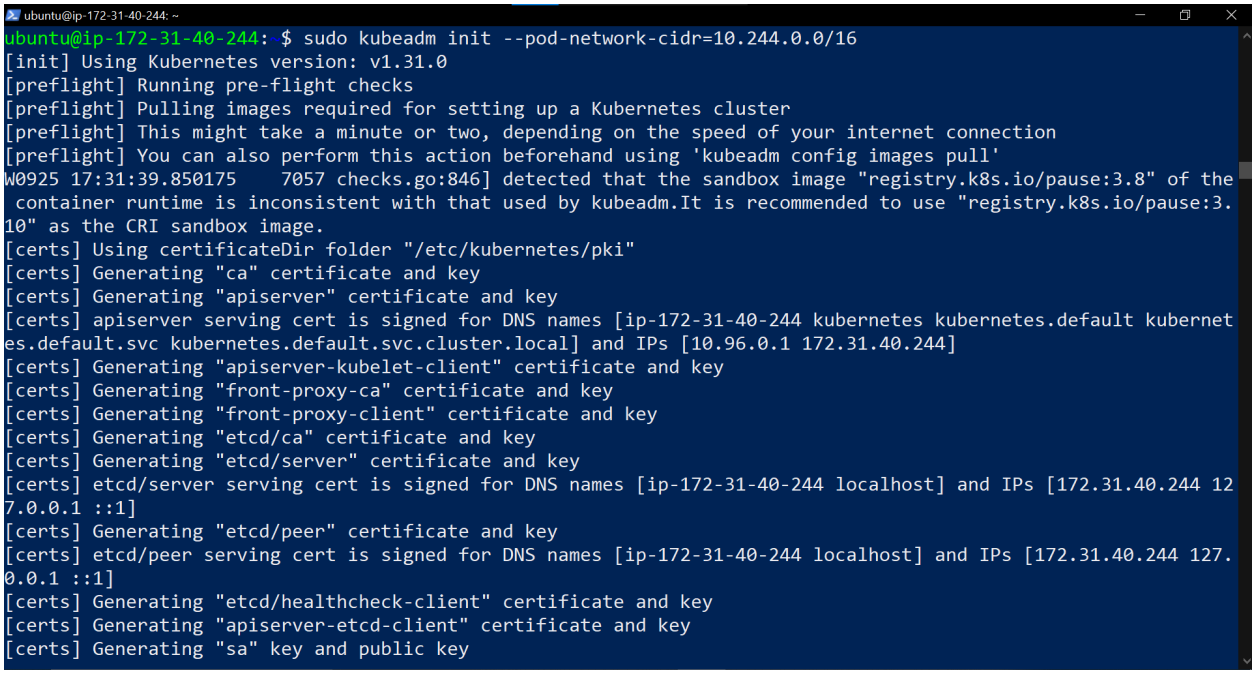
```
ubuntu@ip-172-31-40-244:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-40-244:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-40-244:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Wed 2024-09-25 17:23:54 UTC; 17s ago
       Docs: https://containerd.io
   Main PID: 6555 (containerd)
      Tasks: 7
     Memory: 13.5M (peak: 14.1M)
        CPU: 121ms
     CGroup: /system.slice/containerd.service
             └─6555 /usr/bin/containerd

Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231856399Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231887455Z" level=info msg=servin>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231896683Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231919602Z" level=info msg=servin>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231952908Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231972572Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231980403Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231986601Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 systemd[1]: Started containerd.service - containerd container runtime.
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.234185443Z" level=info msg="conta>
```

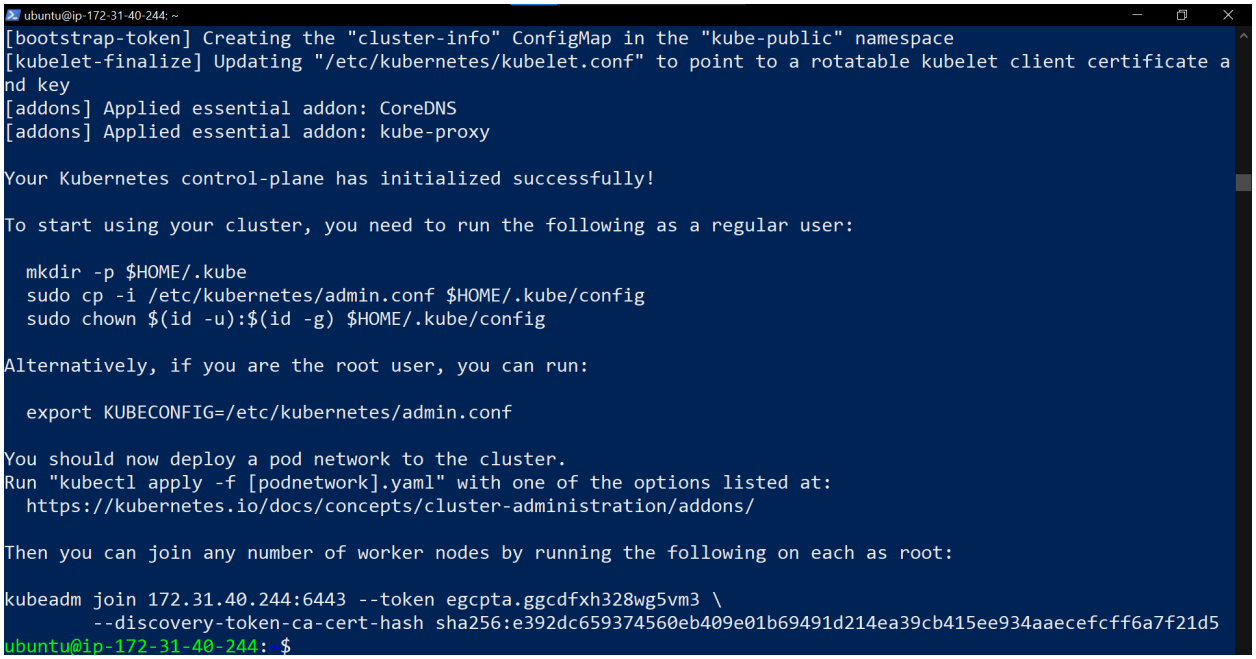
After this, run
sudo apt-get install -y socat

```
ubuntu@ip-172-31-40-244: ~
ubuntu@ip-172-31-40-244:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-ce-rootless-extras libltdl7 libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 142 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (16.8 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.
```

Step 4: Initialize kubecluster:
Sudo kubeadm init --pod-network-cidr=10.244.0.0/16

Name: Mohit Kerkar Div: D15C Roll No: 23





Copy the mkdir and chown commands from the top and execute them.

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

Add a common networking plugin called flannel as mentioned in the code.
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

```
ubuntu@ip-172-31-40-244:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentati
on/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

Now that the cluster is up and running, we can deploy our nginx server on this
Cluster. Apply this deployment file using this command to create a deployment
kubectl apply -f https://k8s.io/examples/application/deployment.yaml

```
ubuntu@ip-172-31-40-244:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-40-244:~$
```

kubectl get pods

```
ubuntu@ip-172-31-40-244:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-brgb7    0/1     Pending   0          48s
nginx-deployment-d556bf558-s5rtj    0/1     Pending   0          48s
ubuntu@ip-172-31-40-244:~$
```

POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80

```
ubuntu@ip-172-31-40-244:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-40-244:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-40-244:~$
```

To untaint all nodes, run the following command
kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-

Then run, kubectl get nodes

```
ubuntu@ip-172-31-40-244:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-40-244 untainted
ubuntu@ip-172-31-40-244:~$ kubectl get nodes
NAME              STATUS   ROLES           AGE   VERSION
ip-172-31-40-244   Ready    control-plane   16m   v1.31.1
ubuntu@ip-172-31-40-244:~$
```

Then run, kubectl get pods command again

```
ubuntu@ip-172-31-40-244:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-brgb7    1/1     Running   0          7m15s
nginx-deployment-d556bf558-s5rtj    1/1     Running   0          7m15s
ubuntu@ip-172-31-40-244:~$
```

Since, Jenkins server/service is already occupying port 8080, i tried starting the nginx service at 8082 port number. Following is the command that i executed was,
kubectl port-forward $POD_NAME 8082:80

```
ubuntu@ip-172-31-40-244:~$ kubectl port-forward svc/nginx 8082:80
Forwarding from 127.0.0.1:8082 -> 80
Forwarding from [::1]:8082 -> 80
Handling connection for 8082
```

The last step consisted of opening a new terminal and logging in using the SSH method that we used earlier and running the following command so as to finally confirm that we have successfully deployed our kubernetes application over the nginx server, using nginx service (opional)

curl --head http://127.0.0.1:8082

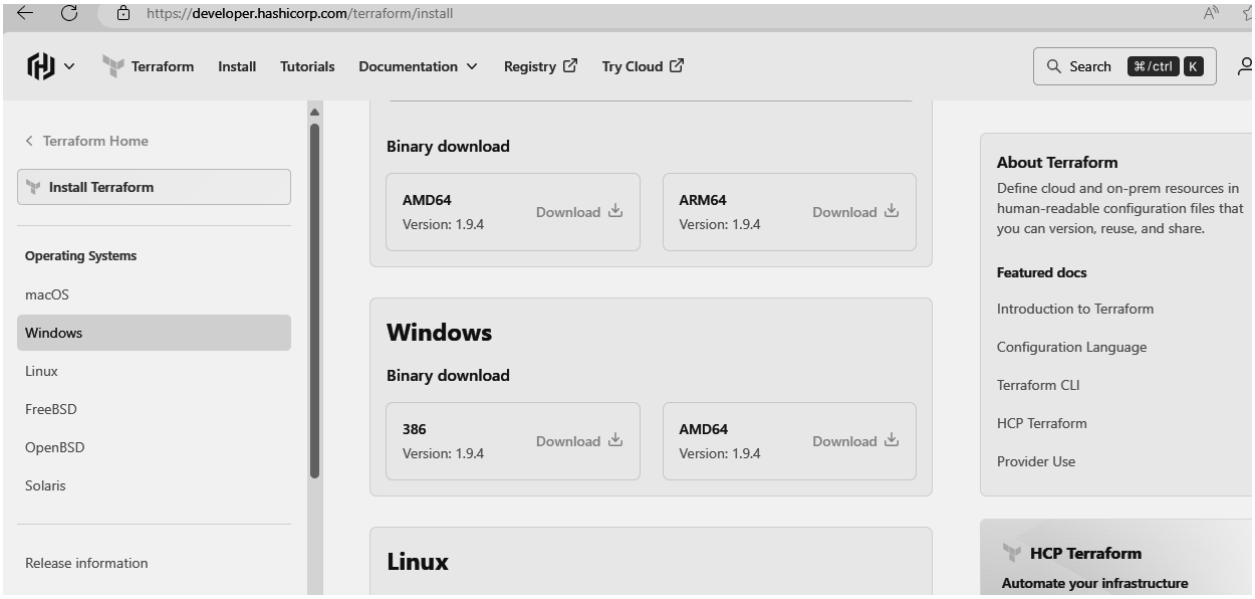
```
Last login: Wed Sep 25 19:13:52 2024 from 45.112.58.76
ubuntu@ip-172-31-40-244:~$ curl --head http://127.0.0.1:8082
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Wed, 25 Sep 2024 19:36:26 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 04 Dec 2018 14:44:49 GMT
Connection: keep-alive
ETag: "5c0692e1-264"
Accept-Ranges: bytes
```

Conclusion:
In this experiment, we successfully deployed an Nginx web server on a Kubernetes cluster and exposed it using NodePort services. By creating the deployment and configuring the service, we demonstrated Kubernetes' ability to manage containerized applications efficiently.
The use of `kubectl` commands allowed us to check pod status and access the Nginx server locally via port forwarding on the EC2 instance. This experience provided valuable insights into service management and networking concepts within Kubernetes.
Overall, this experiment reinforced the fundamental skills needed for deploying cloud-native applications and laid the groundwork for exploring more advanced Kubernetes features in future projects.

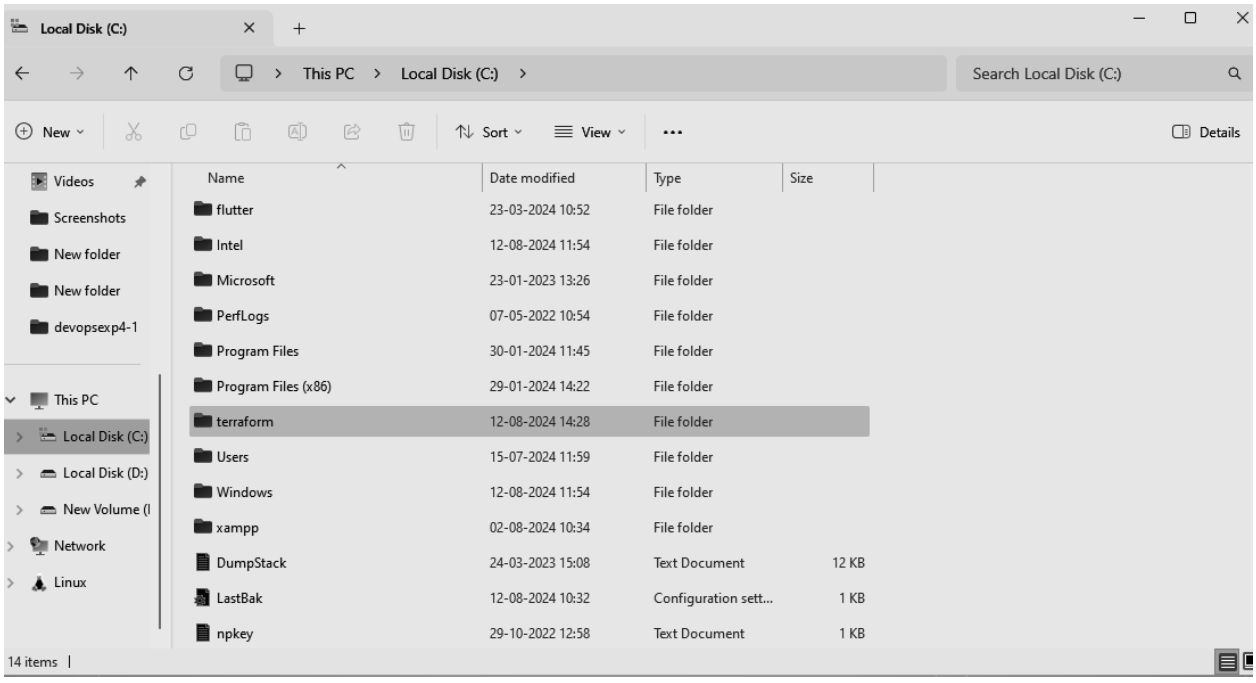
Name: Mohit Kerkar Div: D15C Roll No: 23

Exp 05: To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine.

Step 1: Navigate to hashicorp.com page and download the AMD64 file for Terraform. Install it for Windows.

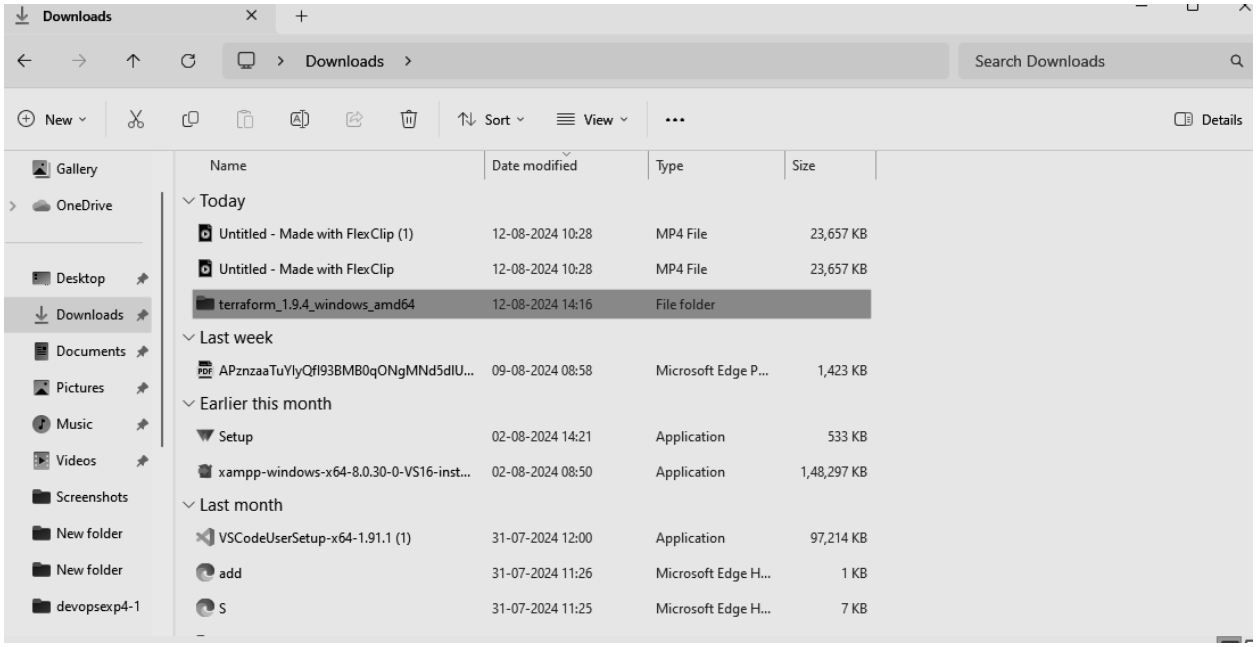


Step 2: Create a new folder inside your C drive named Terraform for your convenience. We would need it for setting up our PATH variable inside system variables.

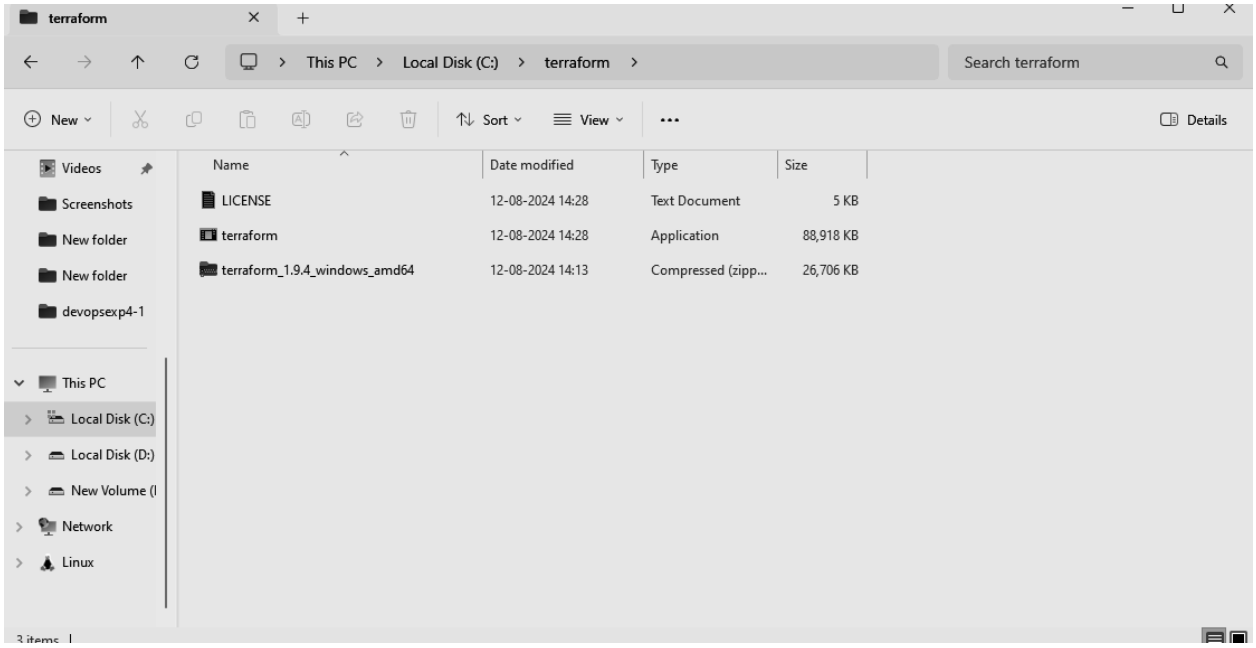


Name: Mohit Kerkar Div: D15C Roll No: 23

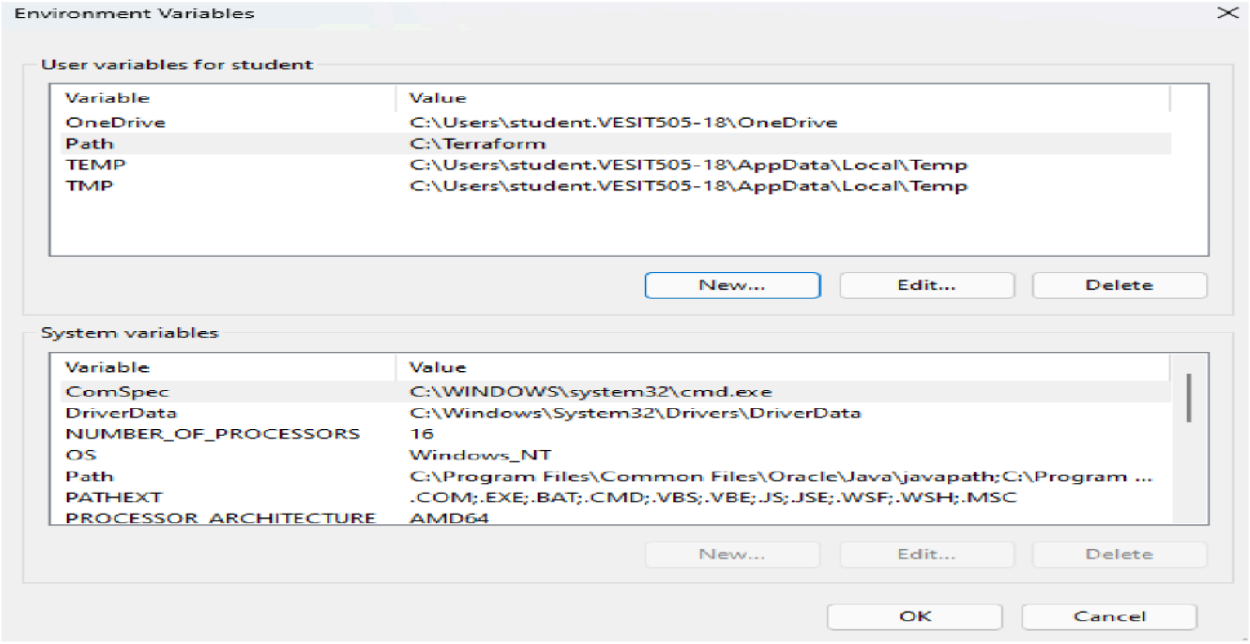
Step 3: Select the package that you have downloaded and extract the files within it. Set it in such a way that it gets extracted in the Terraform folder we created in the step prior to this one



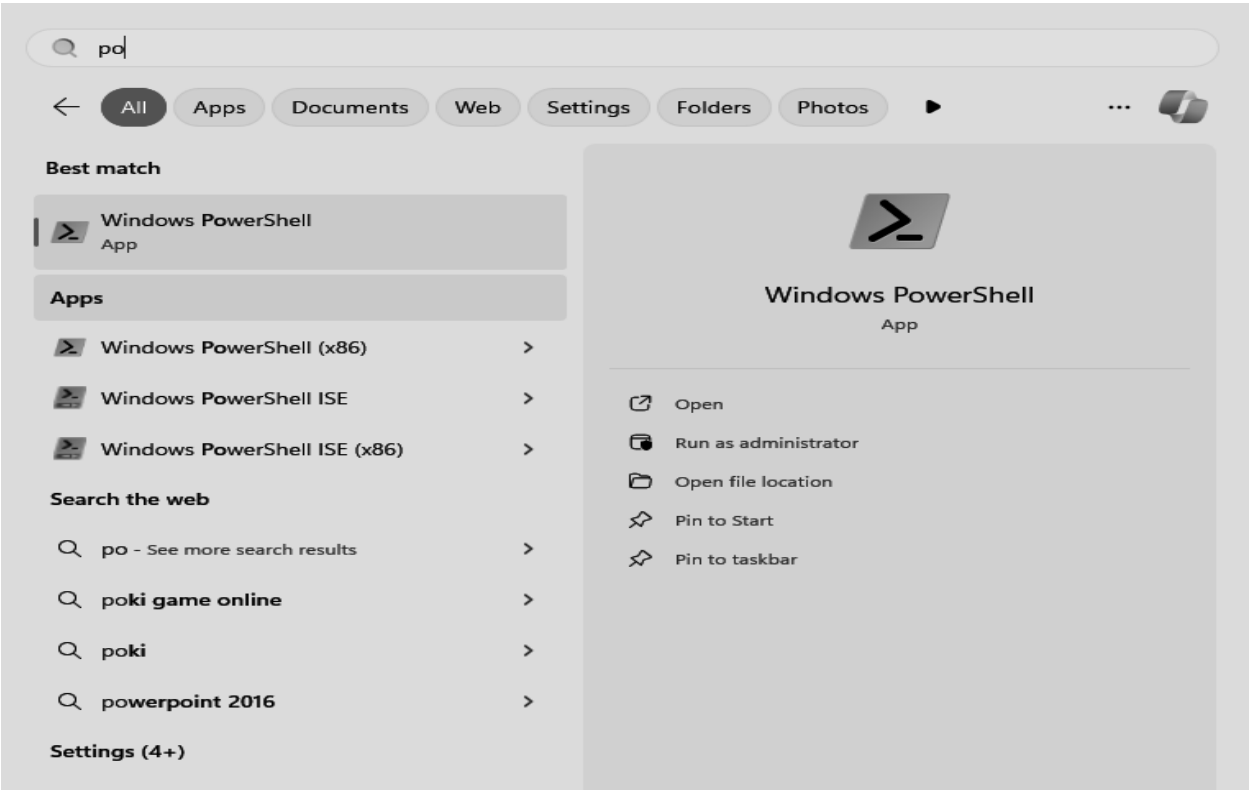
Step 4: Ensure that you have the following files appearing in your newly made Terraform folder in your local C drive



Step 5: Add the path of your Terraform folder in the PATH variable in the system environment variables inside settings.



Step 6: Open Windows PowerShell and select ‘Run as Administrator’ option for carrying out the next step



Step 7: After opening Windows PowerShell, type the command ‘terraform’. If it is properly extracted and path is added to system variables, then following is the result we would be able to see. It will show us numerous other commands and options that we can try out so as to work with Terraform

Conclusion: In conclusion, this experiment walks through the process of installing and setting up Terraform on a Windows machine. By downloading the appropriate package from HashiCorp, extracting it into a designated folder, and adding the folder to the system's PATH variable, we can successfully configure Terraform for use. After completing the installation, running the `terraform` command in PowerShell confirms that the setup is successful, providing access to various Terraform commands. This sets the foundation for managing infrastructure as code using Terraform on a Windows system.

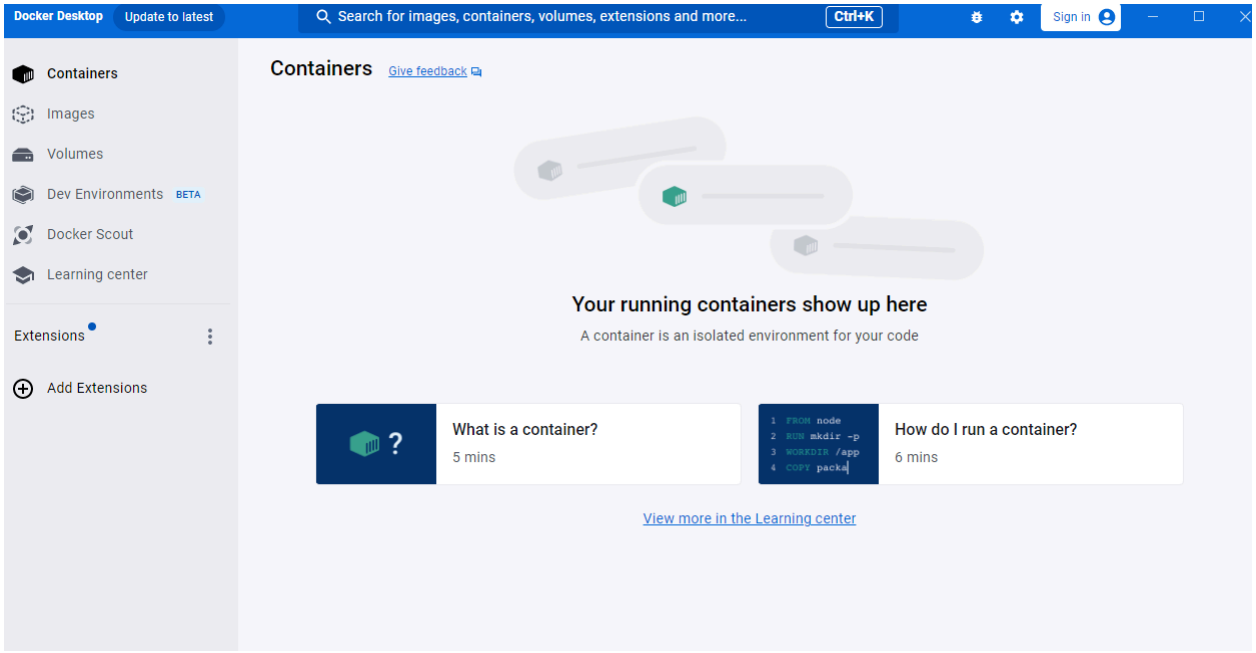
Name: Mohit Kerkar Div: D15C Roll No: 23

Adv DevOps Exp 06

Aim: To create docker image using terraform

Step 1: Check the functionality of docker desktop i.e Check if Docker is installed properly on your system and that Docker Desktop is opening properly (virtualization settings are needed to be enabled in our system bios)

```
C:\Users\INFT505-16>docker -v
Docker version 24.0.6, build ed223bc

C:\Users\INFT505-16>
```



Step 2: Now, create a folder named ‘Terraform Scripts’ in which we save our different types of scripts which will be further used in this experiment.
Step 2: Firstly create a new folder named ‘Docker’ in the ‘TerraformScripts’ folder. Then create a new docker.tf file using Atom editor and write the followingcontents into it to create a Ubuntu Linux container.
Script:
terraform

{ required_providers
{docker = {
source = "kreuzwerker/docker"
version = "2.21.0"

```
}
}
}
provider "docker" {
host = "npipe:////.//pipe//docker_engine"
}
# Pulls the image
resource "docker_image" "ubuntu"
{name = "ubuntu:latest"
}
# Create a container
resource "docker_container" "foo"
{ image =
docker_image.ubuntu.image_idname =
"foo"
}
```

Step 3: Run terraform init command

Run the command **terraform init** in your terminal. This command initializes the Terraform configuration, downloads the necessary provider plugins specified in the `docker.tf` file, and prepares the working directory for further commands. This step is crucial to ensure that Terraform can interact with Docker.

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of kreuzwerker/docker from the dependency lock file
- Using previously-installed kreuzwerker/docker v2.21.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>
```

Step 4: Run terraform plan command:

Next, execute the command **terraform plan** to create an execution plan. This command will show what actions Terraform will take based on your configuration. It ensures that the planned changes match your expectations before actual implementation.

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = [
          + "sleep",
          + "3600",
        ]
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = (known after apply)
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
      + name             = "foo"
      + network_data     = (known after apply)
      + read_only        = false
      + remove_volumes   = true
```

```
      + rm               = false
      + runtime          = (known after apply)
      + security_opts    = (known after apply)
      + shm_size         = (known after apply)
      + start            = true
      + stdin_open       = false
      + stop_signal      = (known after apply)
      + stop_timeout     = (known after apply)
      + tty              = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

  # docker_image.ubuntu will be created
  + resource "docker_image" "ubuntu" {
      + id          = (known after apply)
      + image_id    = (known after apply)
      + latest      = (known after apply)
      + name        = "ubuntu:latest"
      + output      = (known after apply)
      + repo_digest = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.

─────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if
you run "terraform apply" now.

C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>
```

Step 5: Execute terraform apply command:
Run **terraform apply** to apply the configuration and create the specified resources. During this step, Terraform will prompt you to confirm the action by typing yes. Upon confirmation, Terraform will pull the Ubuntu image and create the container, reflecting successful deployment.

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated wi
  + create

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = [
          + "sleep",
          + "3600",
        ]
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = (known after apply)
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
      + name             = "foo"
      + network_data     = (known after apply)
      + read_only        = false
      + remove_volumes   = true
```

We are supposed to put yes to proceed ahead

```
      + rm               = false
      + runtime          = (known after apply)
      + security_opts    = (known after apply)
      + shm_size         = (known after apply)
      + start            = true
      + stdin_open       = false
      + stop_signal      = (known after apply)
      + stop_timeout     = (known after apply)
      + tty              = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

  # docker_image.ubuntu will be created
  + resource "docker_image" "ubuntu" {
      + id          = (known after apply)
      + image_id    = (known after apply)
      + latest      = (known after apply)
      + name        = "ubuntu:latest"
      + output      = (known after apply)
      + repo_digest = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

After executing terraform apply:

```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

docker_image.ubuntu: Creating...
docker_image.ubuntu: Still creating... [10s elapsed]
docker_image.ubuntu: Creation complete after 12s [id=sha256:b1e9cef3f2977f8bdd19eb9ae04f83b315f80fe4f5c5651fedf41482c12432f7ubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 1s [id=684a1de22619abba495d0c2886fc9d8f681302574f3e1ca16c5297470b4e88b4]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>
```

Terraform images after apply step:

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
ubuntu       latest    b1e9cef3f297   3 weeks ago     78.1MB
sonarqube    latest    3183d6818c6e   11 months ago   716MB

C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>
```

Step 6: Run terraform destroy command
To remove the resources created during the experiment, run the command **terraform destroy.** This command will prompt for confirmation before proceeding to delete the container and any other associated resources. It's essential for maintaining a clean working environment and avoiding unnecessary resource consumption.

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:b1e9cef3f2977f8bdd19eb9ae04f83b315f80fe4f5c5651fedf41482c12432f7ubuntu:latest]
docker_container.foo: Refreshing state... [id=684a1de22619abba495d0c2886fc9d8f681302574f3e1ca16c5297470b4e88b4]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach            = false -> null
      - command           = [
          - "sleep",
          - "3600",
        ] -> null
      - cpu_shares        = 0 -> null
      - dns               = [] -> null
      - dns_opts          = [] -> null
      - dns_search        = [] -> null
      - entrypoint        = [] -> null
      - env               = [] -> null
      - gateway           = "172.17.0.1" -> null
      - group_add         = [] -> null
      - hostname          = "684a1de22619" -> null
      - id                = "684a1de22619abba495d0c2886fc9d8f681302574f3e1ca16c5297470b4e88b4" -> null
      - image             = "sha256:b1e9cef3f2977f8bdd19eb9ae04f83b315f80fe4f5c5651fedf41482c12432f7" -> null
      - init              = false -> null
      - ip_address        = "172.17.0.2" -> null
      - ip_prefix_length  = 16 -> null
      - ipc_mode          = "private" -> null
      - links             = [] -> null
      - log_driver        = "json-file" -> null
      - log_opts          = {} -> null
```

After executing terraform destroy command:

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
sonarqube    latest    3183d6818c6e   11 months ago   716MB

C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>
```

Conclusion: In this experiment, we effectively utilized Terraform to manage Docker resources by creating an organized folder structure and executing a series of commands to deploy an Ubuntu container. By following the steps from initialization to resource destruction, we gained hands-on experience in infrastructure as code (IaC) principles, enhancing our understanding of managing containers with Terraform and Docker. This practical approach not only solidified our knowledge but also emphasized the importance of automated resource management in modern DevOps practices.

Name: Mohit Kerkar Div: D15C Roll No: 23

Adv DevOps Exp 07

Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Step 1:
Ensure Installations and path settings of the following softwares/services:

1. **Jenkins**
2. **Docker Desktop Engine**

Both these services help us create a sonarqube image and build a project within sonarqube using the pipeline architecture in jenkins.

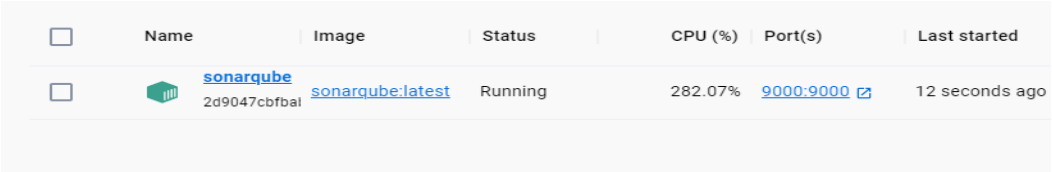
Run this command in powershell administrator mode: docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

This kickstarts the Sonarqube service/server at post **9000**. It also creates a Docker image within our Docker Desktop account. Make sure to keep your Docker Engine up and running

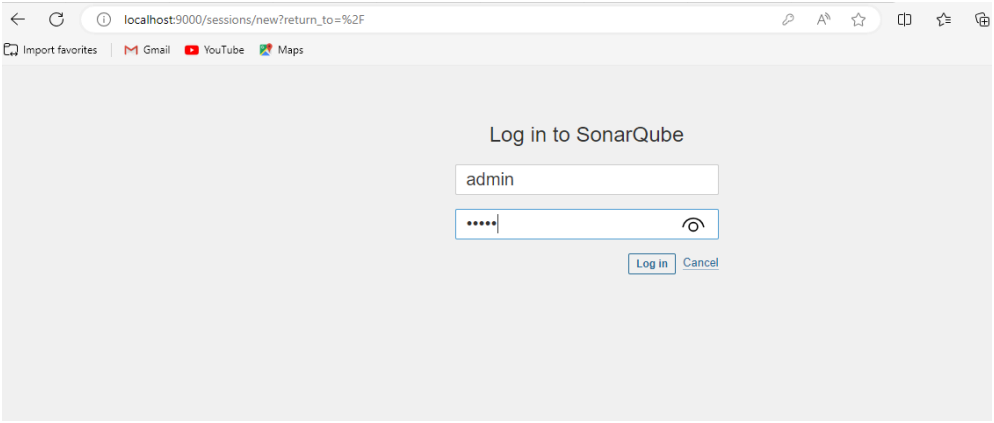
Powershell output:

```
PS C:\Users\INFT505-16> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:
latest
2d9047cbfbab55bd03ed72f7349feebf5d79434b23ff7eda329694bd24115349
PS C:\Users\INFT505-16>
```

Sonarqube started on Docker:

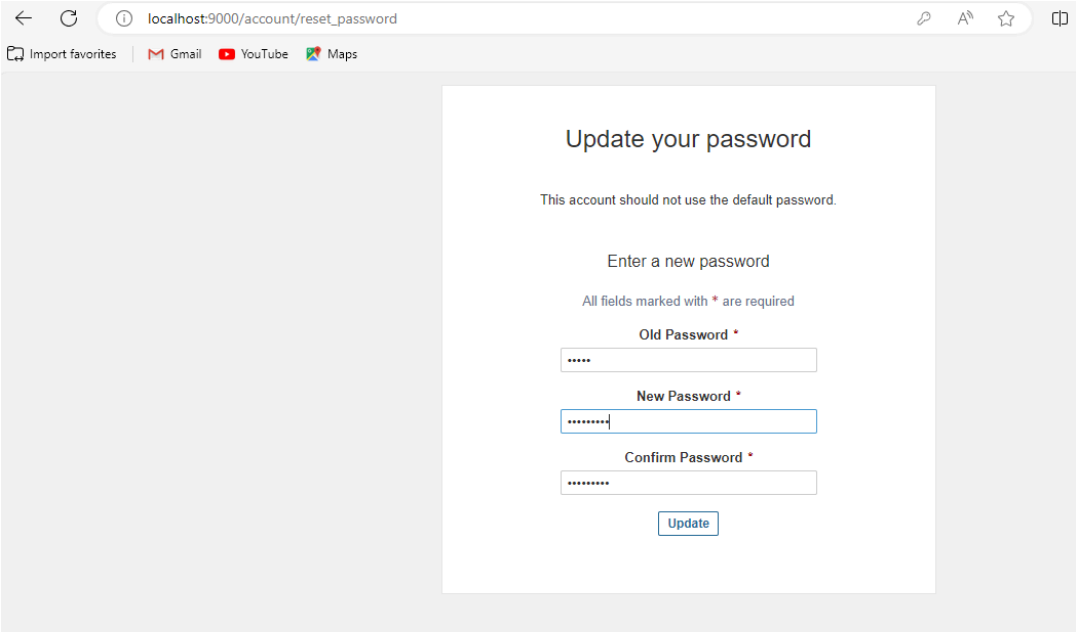


Go to localhost:9000. On fulfilling the desired action, we arrive on a page that asks us for our username and password.
The default credentials for Sonarqube server are (username,password) == **(admin,admin)**



Name: Mohit Kerkar Div: D15C Roll No: 23

Sonarqube asks us to update the default password and make a new password for this Sonarqube account. So i entered a new suitable password (keep it easy enough for one to remember)

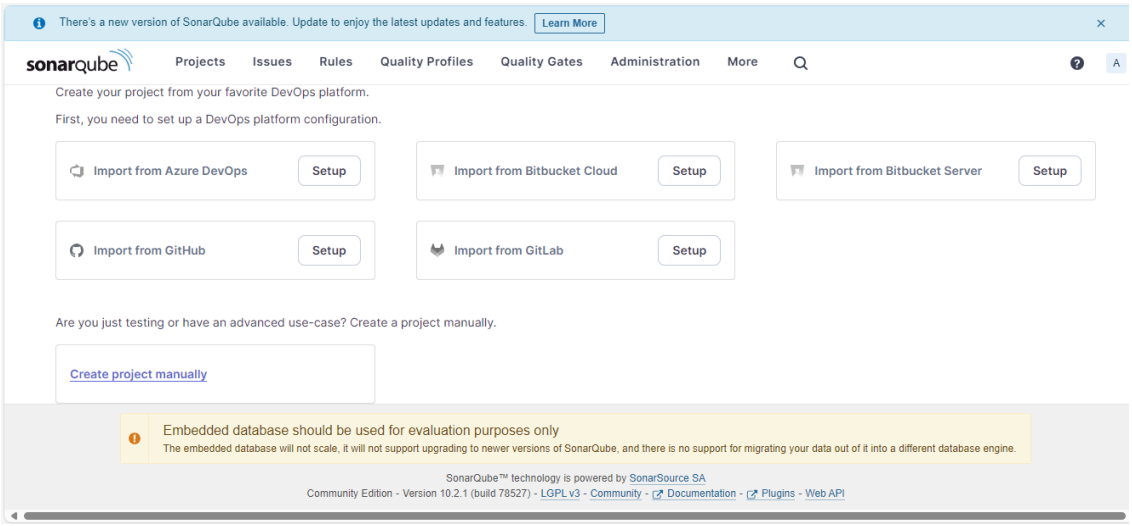


Step 2:

After logging in,

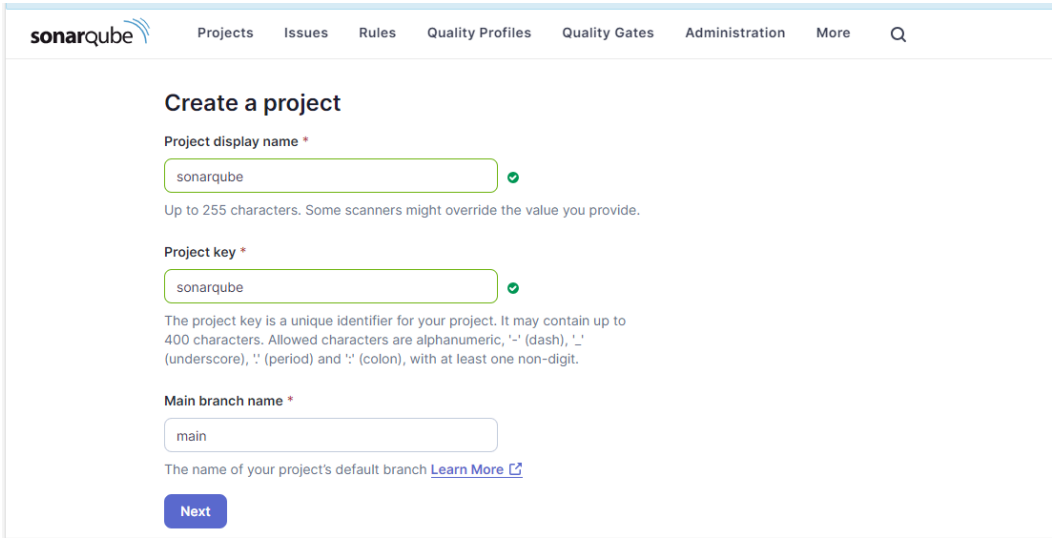
This is the page that we land on… This tells us that with the help of docker we have successfully created a Sonarqube image and that we have assigned a port number 9000 to it so as to use its services and build a Sonarqube project.

From here on, we are supposed to make a new project manually by selecting the ‘Create project manually’ option

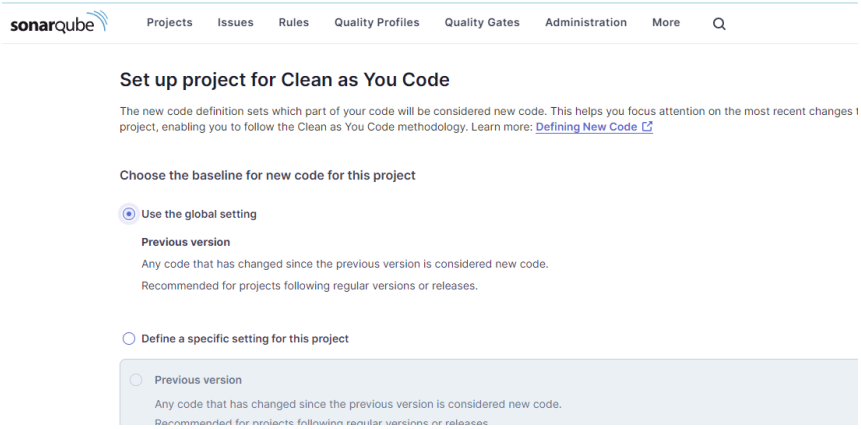


Name: Mohit Kerkar Div: D15C Roll No: 23

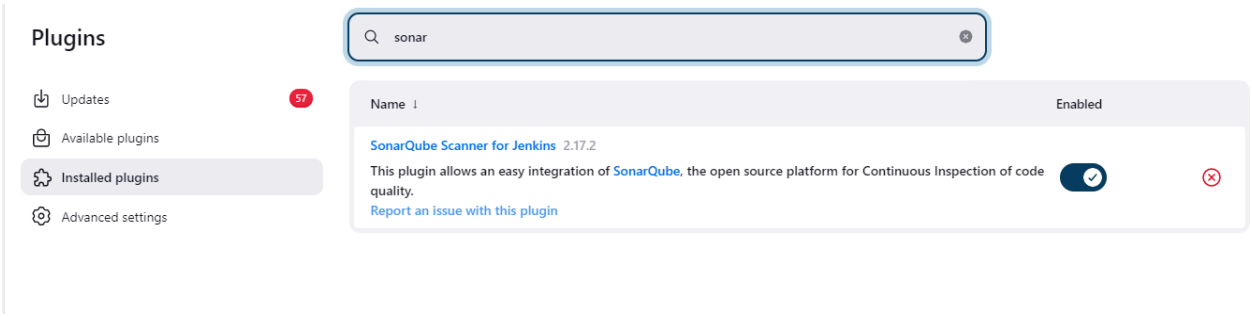
We are now supposed to enter the name for our Sonarqube
Lets keep it as ‘sonarqube’



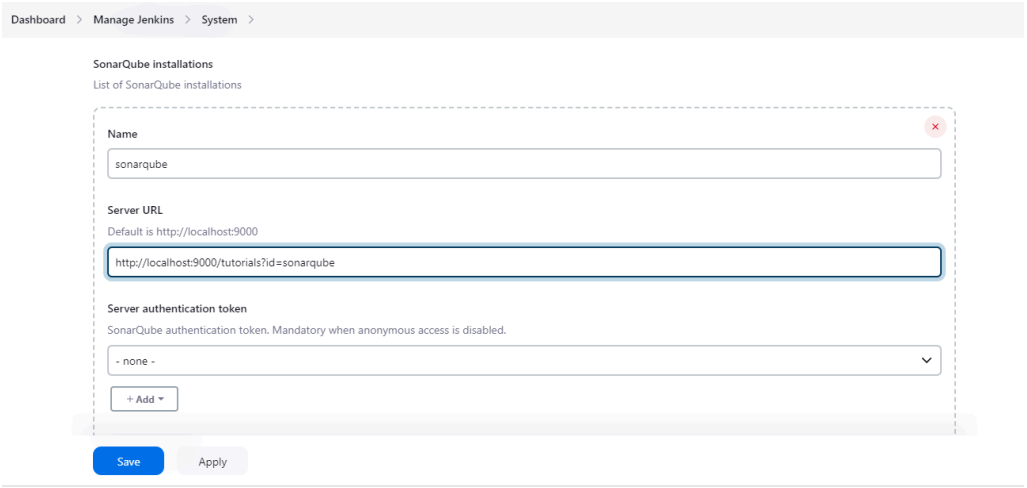
Next, we are asked for setting up our sonarqube project
We will select global settings for this project and finally create the project named ‘sonarqube’



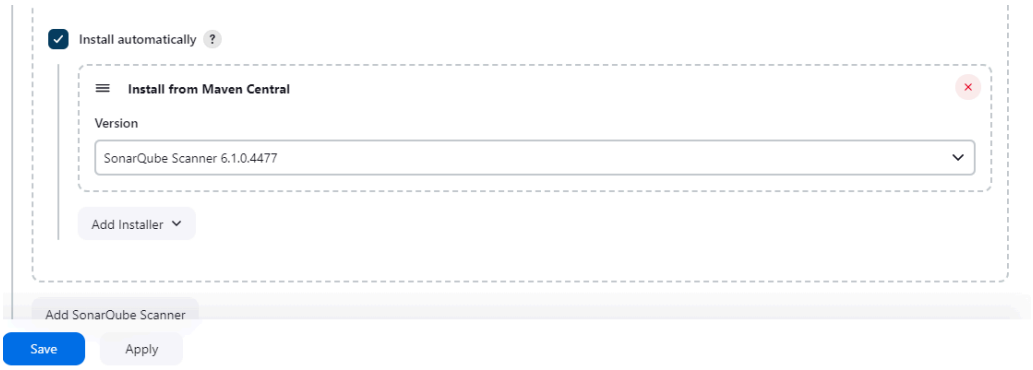
Step 3: Go to Jenkins Dashboard and download the plugin named **‘Sonarqube Scanner for Jenkins’**. This plugin will help us to build our Sonarqube project through a project created within jenkins. Make sure to restart Jenkins once installation is complete and Enable this plugin to use its services



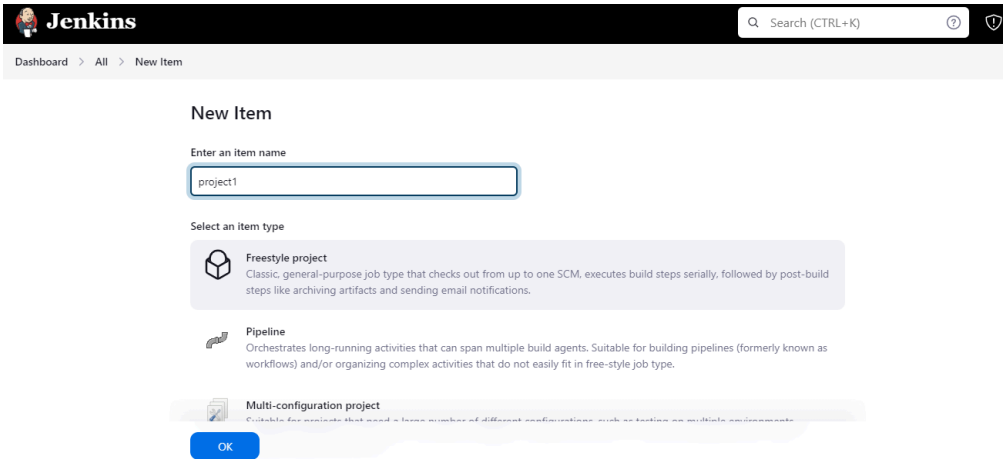
Go to jenkins system configurations, and navigate to Sonarqube installation
Enter the Sonarqube's project name, the URL and server authentication token if needed



Select 'install automatically' option for installation of the specified Sonnarqube Scanner version

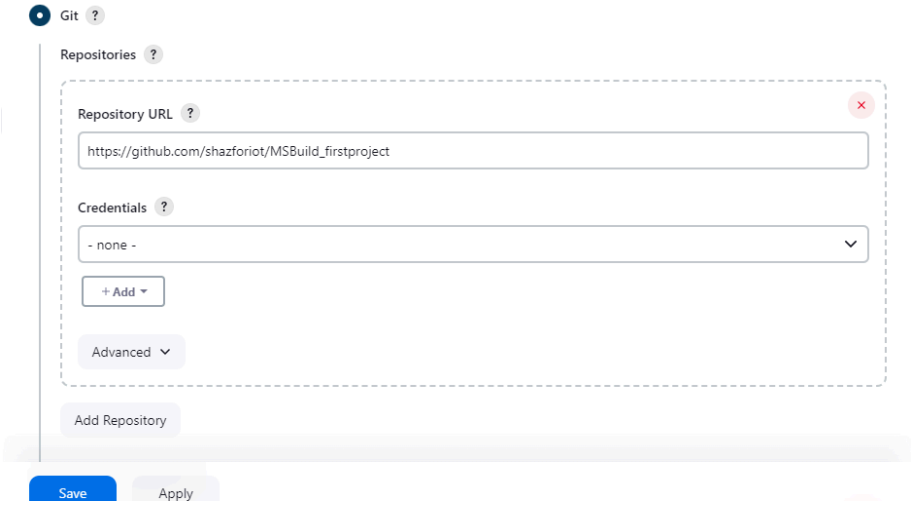


Step 4: Make a freestyle project in jenkins and give it a suitable name. This project is being made by us, so as to carry out our build of our Sonarqube project and perform static analysis of it based on whether the build is successful or not.

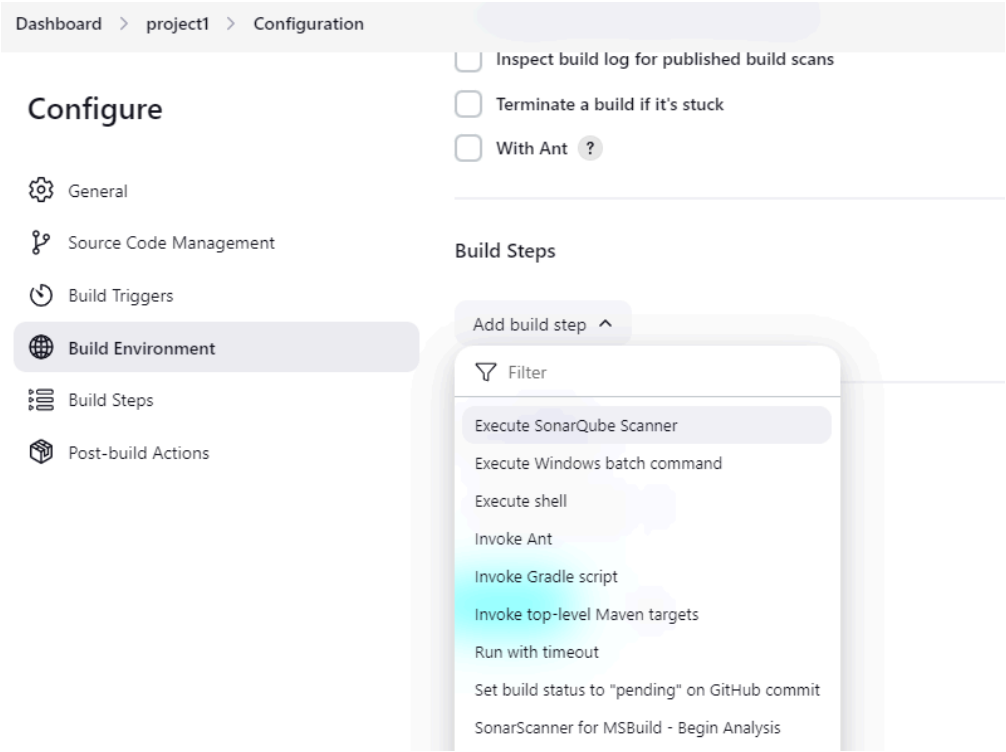


Name: Mohit Kerkar Div: D15C Roll No: 23

Choose this GitHub repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject.git
It is a sample hello-world project with no vulnerabilities and issues, just to test



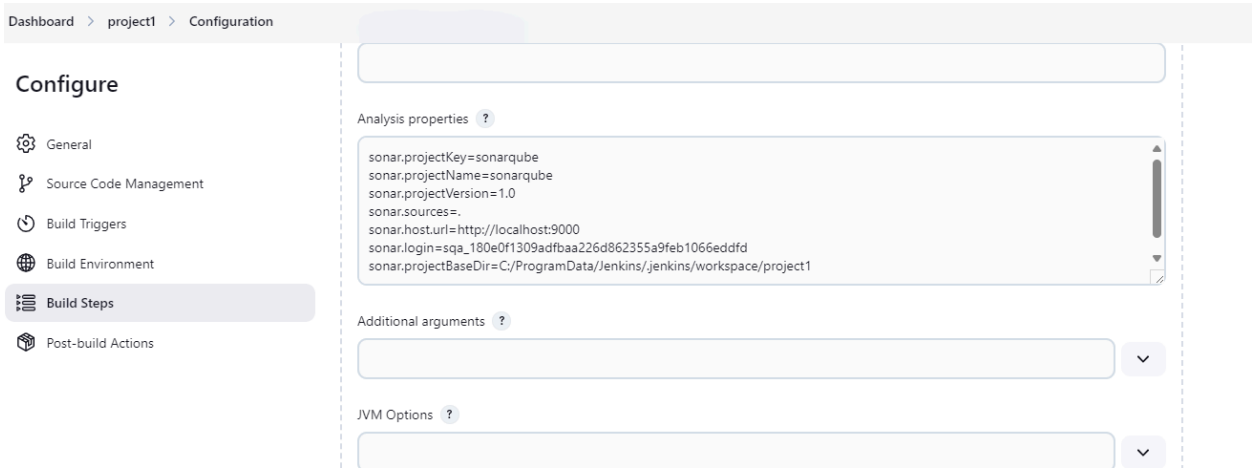
Next, under build steps, select ‘Execute Sonarqube Scanner’ option



Name: Mohit Kerkar Div: D15C Roll No: 23

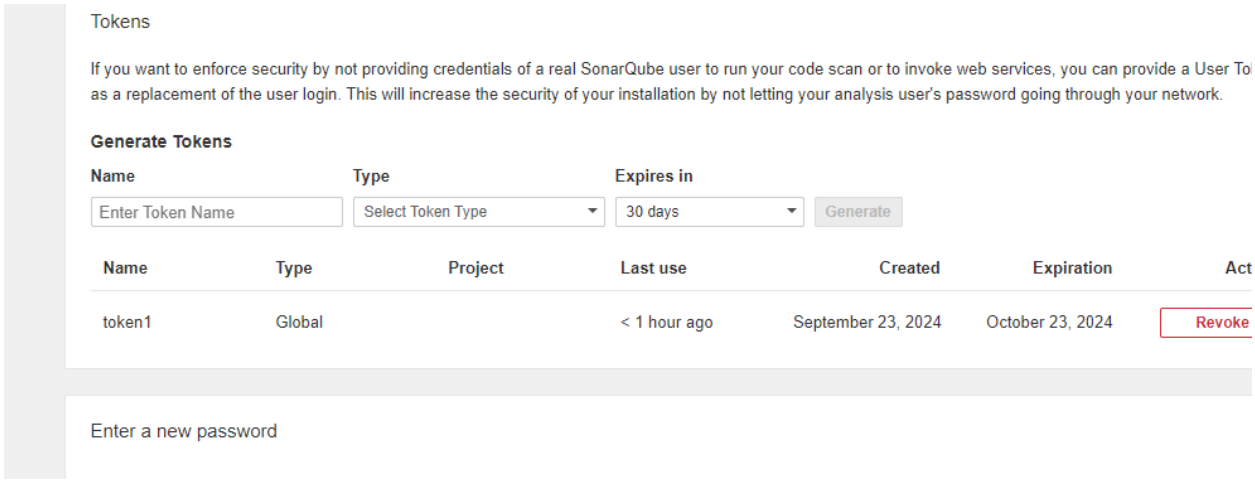
There is an option for entering ‘Analysis properties’ in which i entered the following credentials/parameters related to my Sonarqube account or the project that i have made:

sonar.projectKey=com.example.myproject
sonar.projectName=My Sample Project
sonar.projectVersion=1.0
sonar.sources=.
sonar.host.url=http://localhost:9000
sonar.login=sqa_180e0f1309adfbaa226d862355a9feb1066eddfd
sonar.projectBaseDir=C:/ProgramData/Jenkins/.jenkins/workspace/project1



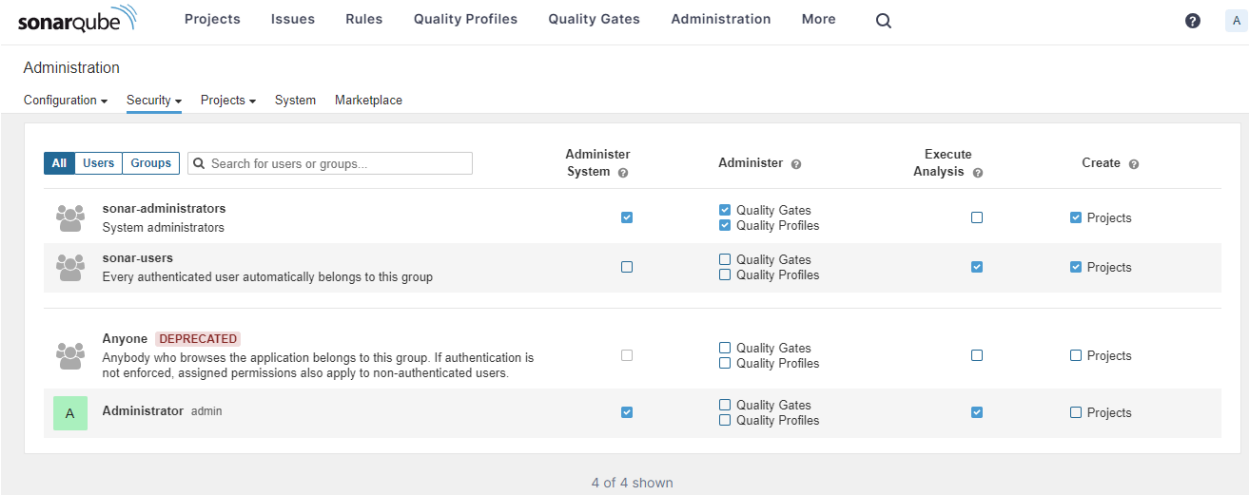
Step 5: In the analysis properties, in order to set the **sonar.login** parameter, we are supposed to navigate to our account details and choose the option of generating a global token. Create a new token…
Now, this assists us to identify our project uniquely without having to provide our credentials or in case our credentials fail to be identified

This token creation step is optional and is only to be used when our analysis properties consisting of our normal credentials fail

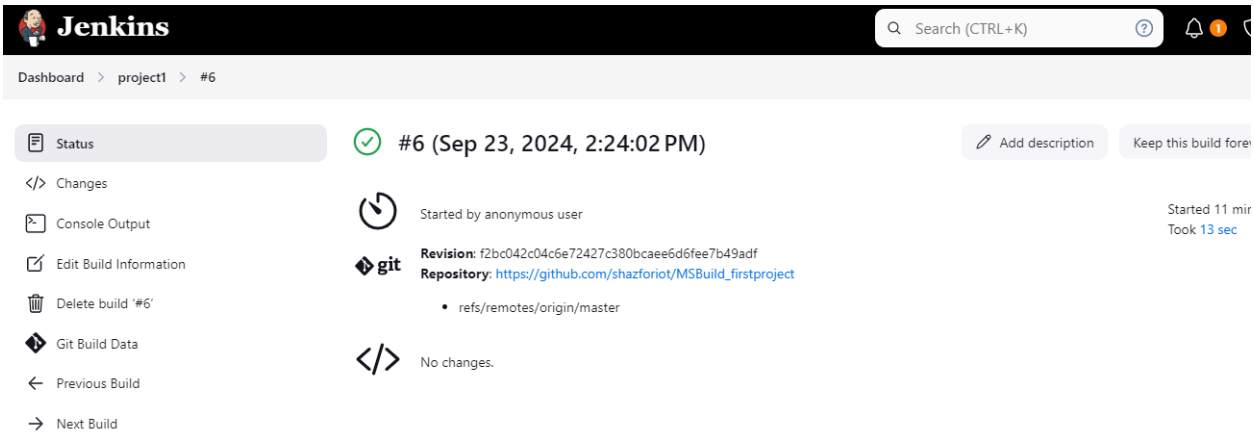


Name: Mohit Kerkar Div: D15C Roll No: 23

Now, navigate to security within administration and provide administrator system and execute analysis rights to the entity named admin.



After Building…the console output should indicate success
This indicates that the image that we created in docker successfully led us to creating a project within sonarqube, which in turn led us to building it over jenkins by uniquely identifying our sonarqube project. Build stage of our sonarqube project in jenkins involves static analysis for which we passed in the appropriate properties



Conclusion: The integration of Jenkins with SonarQube for static analysis using Docker provides a streamlined and automated approach to ensuring code quality. By successfully configuring the necessary tools—Docker for hosting SonarQube and Jenkins for building and analyzing the project—you can perform seamless static analysis using SonarQube's powerful scanning capabilities. This setup automated vulnerability detection, ensuring that potential issues are caught early. The use of GitHub as a repository for code allows for real-time scanning and continuous integration, ensuring that projects meet quality standards before moving forward in the development lifecycle.

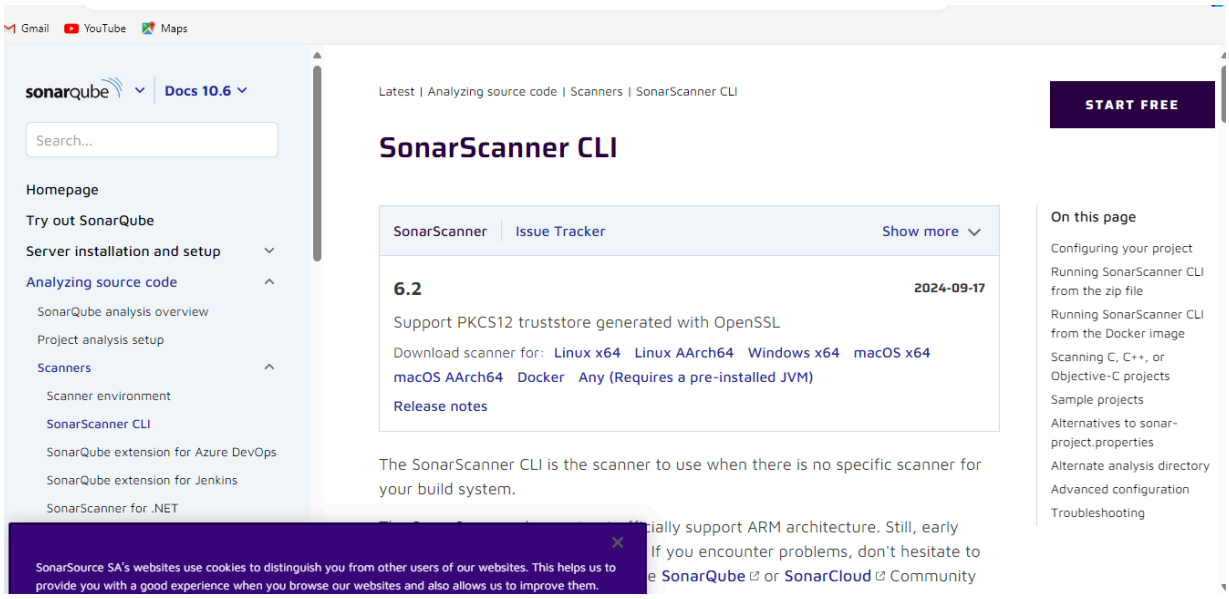
Name: Mohit Kerkar Div: D15C Roll No: 23

Adv DevOps Exp 08

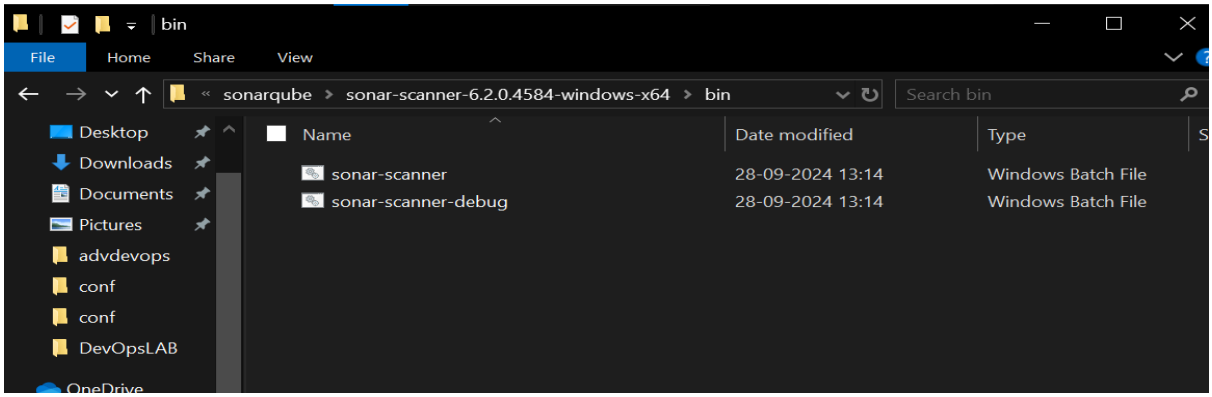
Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web /Java / Python application.
(I have performed this experiment entirely on my laptop)

For the Docker installation problems that i was facing during experiments 6 and 7, I resorted to the option of Resetting Windows which helped me clear the corrupted files which were causing my Laptop's Restart issues

Step 1: Download sonar scanner. We will require it in the further steps while building our pipeline
SonarScanner CLI (sonarsource.com)
Visit this link and download the sonarqube scanner CLI.

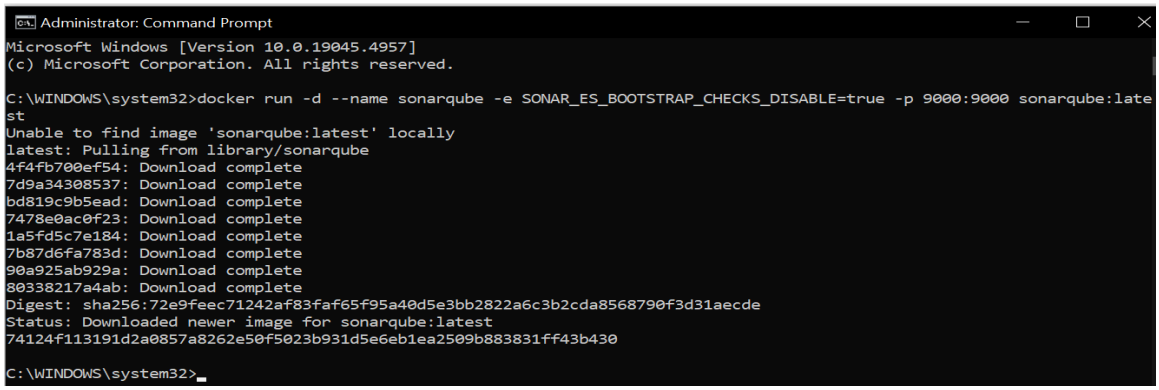


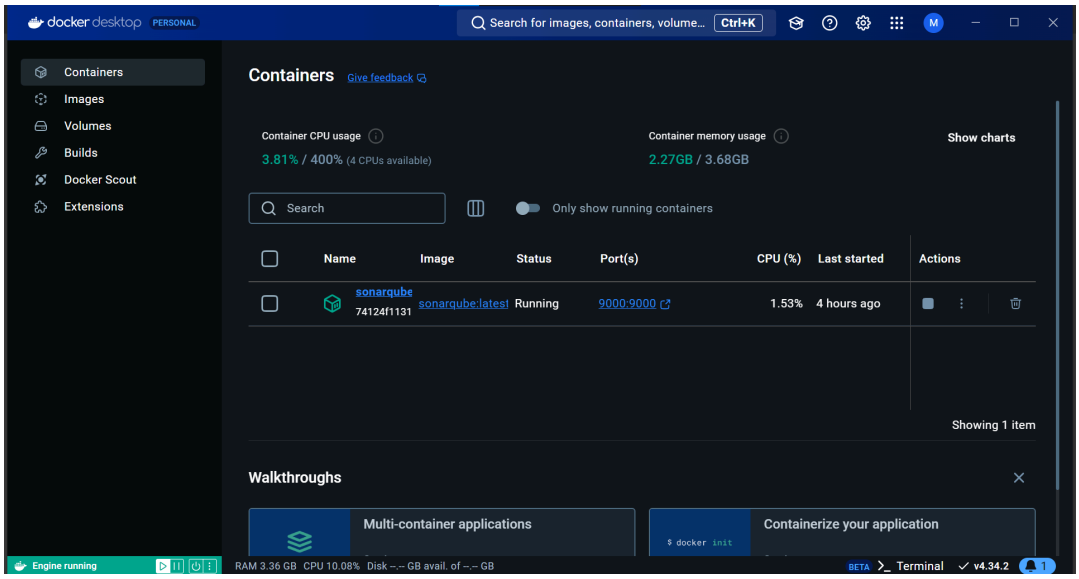
Extract the zip files in a folder



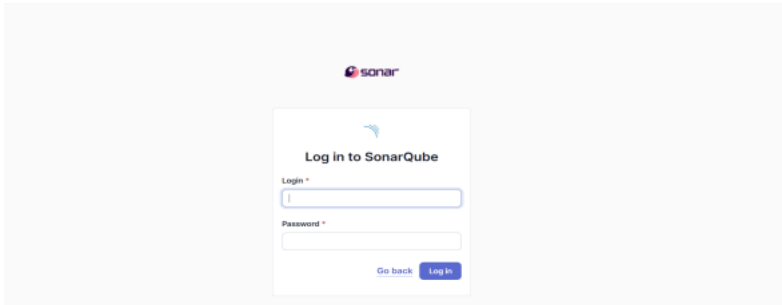
Name: Mohit Kerkar Div: D15C Roll No: 23

Run this command in an administrative command prompt so as to create a sonarqube image on localhost:9000



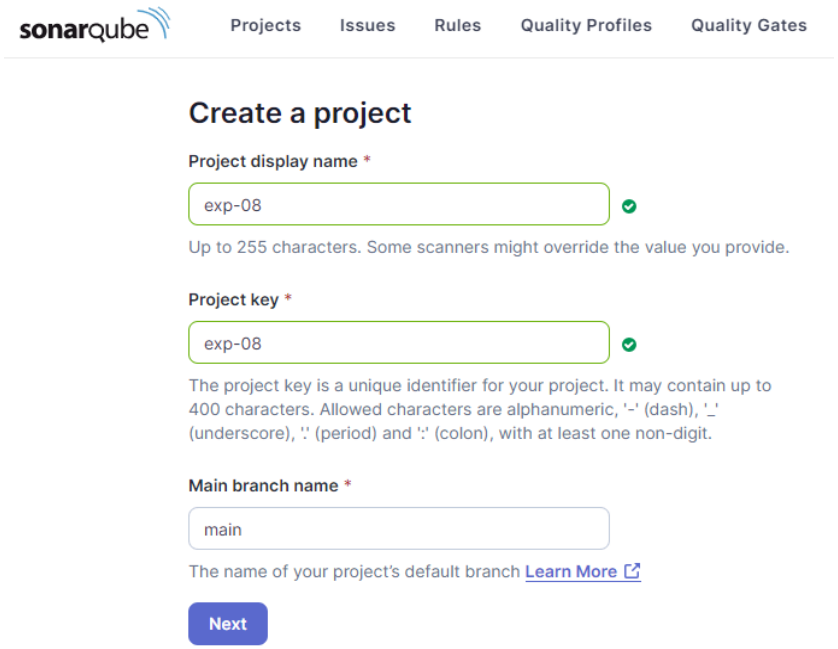


Visit localhost:9000 on your browser and login to your account(the default credentials are admin and admin for username and password. It allows to change it and set our own password for our account)

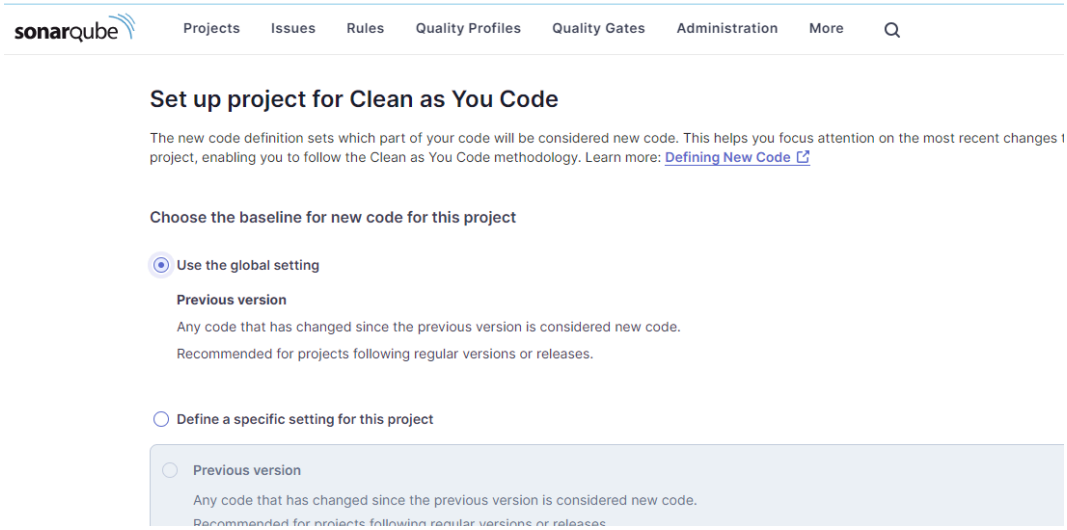


Next, we are supposed to Create a project. Give some suitable name to your project. A project will be generated for it automatically. We can change it or keep it the same

Name: Mohit Kerkar Div: D15C Roll No: 23



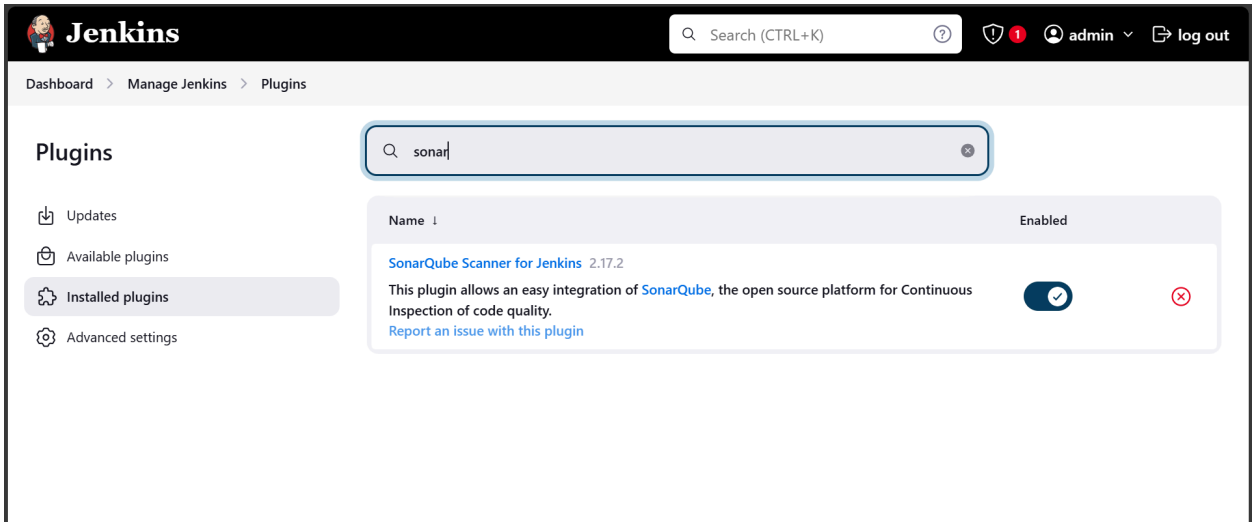
Choose global settings option for setting up your project



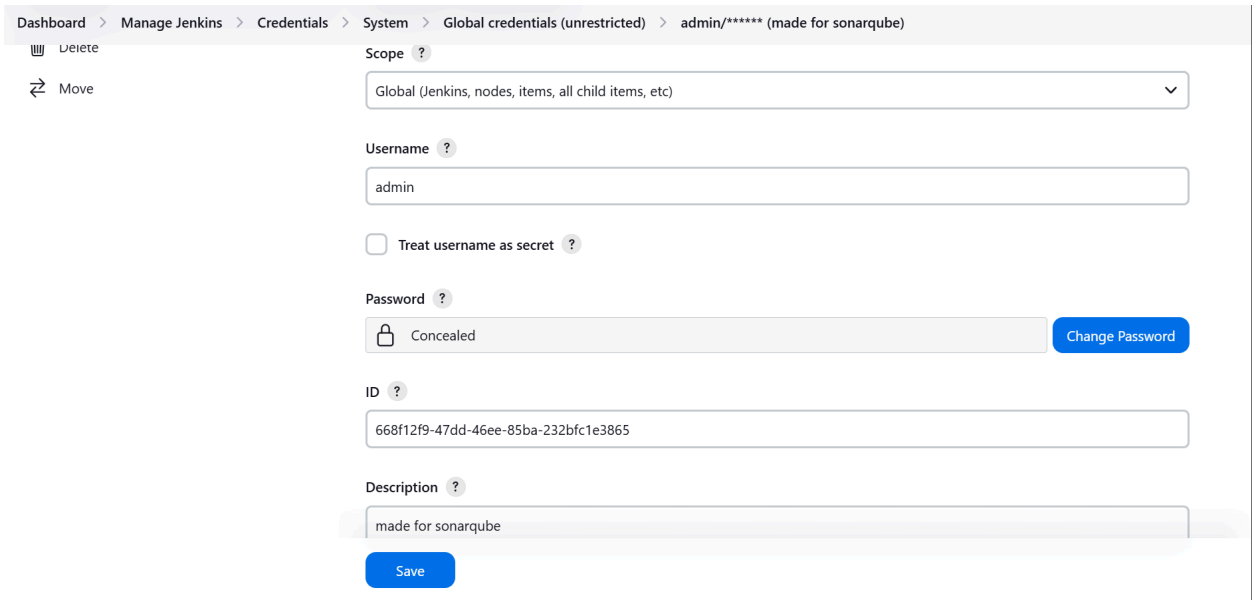
Now, on jenkins ..
For a few contents inside the script to be built using our pipeline, we need to configure those changes inside the global configuration settings inside Manage Jenkins.

First, Go to plugins inside Manage Jenkins and install the Sonarqube Scanner plugin which is necessary for the build to be executed.

Name: Mohit Kerkar Div: D15C Roll No: 23



Next, we will have to add our Sonarqube project specific credentials in the Credentials section of manage jenkins

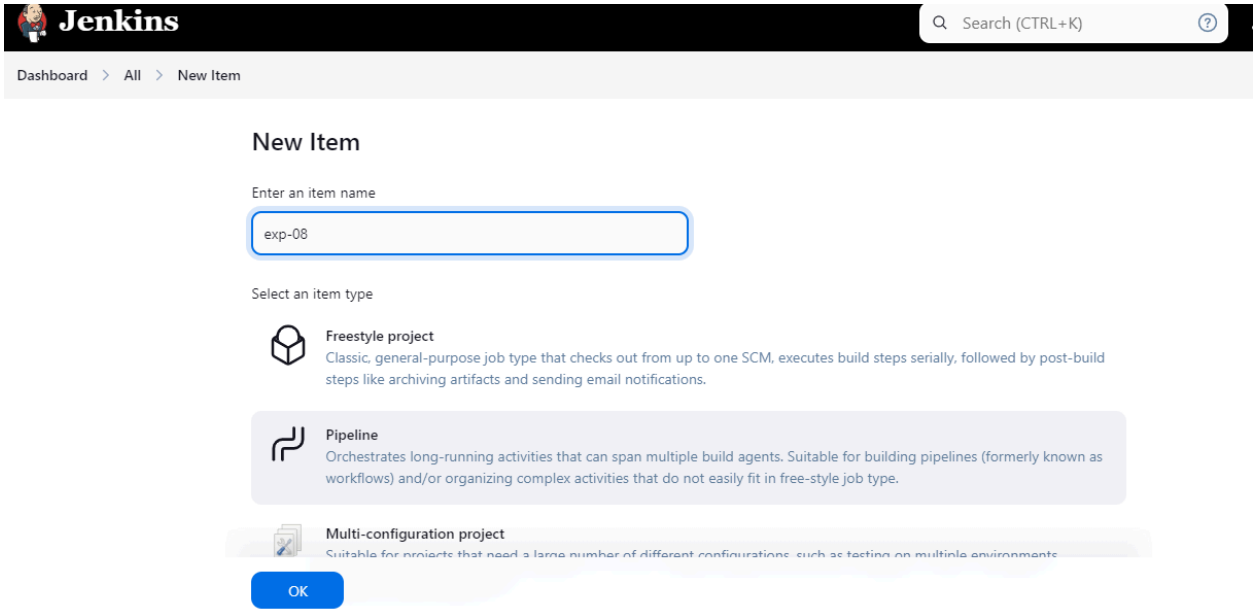


Here, i added the username i.e admin and my password … for which jenkins generated a automated ID for the credential block.

Now, our pipeline script will contain a line where we are supposed to provide the build with our source directory of the sonar-scanner batch file which we installed in the initial steps of our experiment
C:\sonarqube\sonar-scanner-6.2.0.4584-windows-x64\bin\sonar-scanner.bat
We will add this path inside the script to be built

Now, come to the dashboard
Create new item and select pipeline project. Give it a suitable name

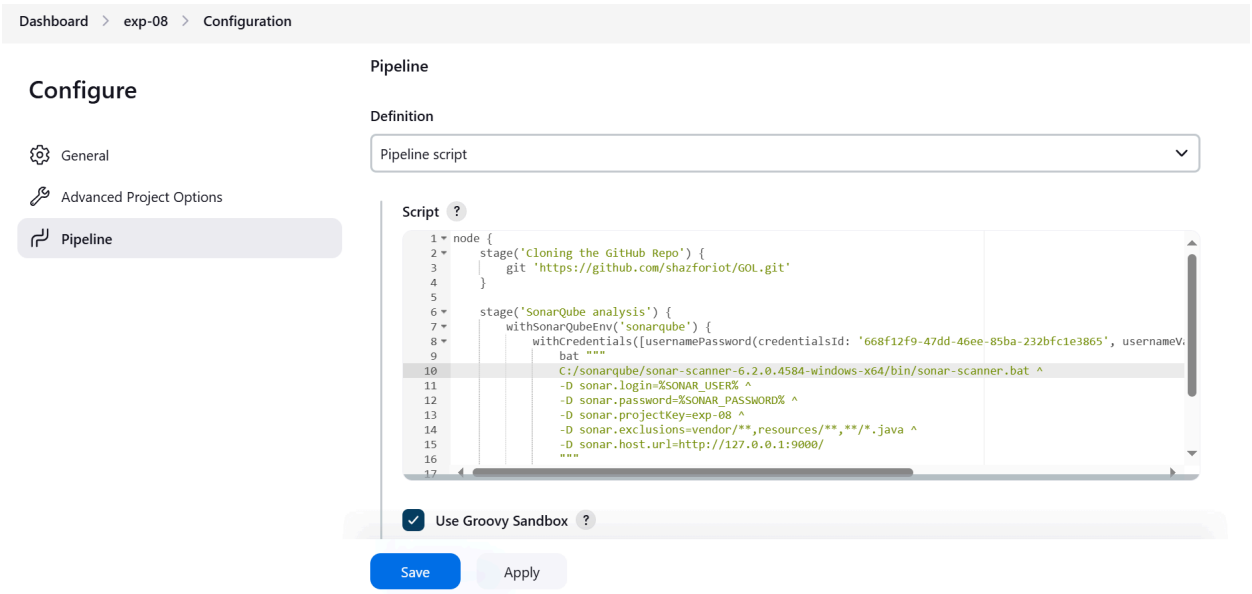
Name: Mohit Kerkar Div: D15C Roll No: 23



Put this script in the Pipeline Definition section and select the mode as Pipeline script

```
node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/shazforiot/GOL.git'
    }

    stage('SonarQube analysis') {
        withSonarQubeEnv('sonarqube') {
            withCredentials([usernamePassword(credentialsId:
'668f12f9-47dd-46ee-85ba-232bfc1e3865', usernameVariable: 'SONAR_USER',
passwordVariable: 'SONAR_PASSWORD')]) {
                bat """
                C:/sonarqube/sonar-scanner-6.2.0.4584-windows-x64/bin/sonar-scanner.bat ^
                -D sonar.login=%SONAR_USER% ^
                -D sonar.password=%SONAR_PASSWORD% ^
                -D sonar.projectKey=exp-08 ^
                -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
                -D sonar.host.url=http://127.0.0.1:9000/
                """
            }
        }
    }
}
```

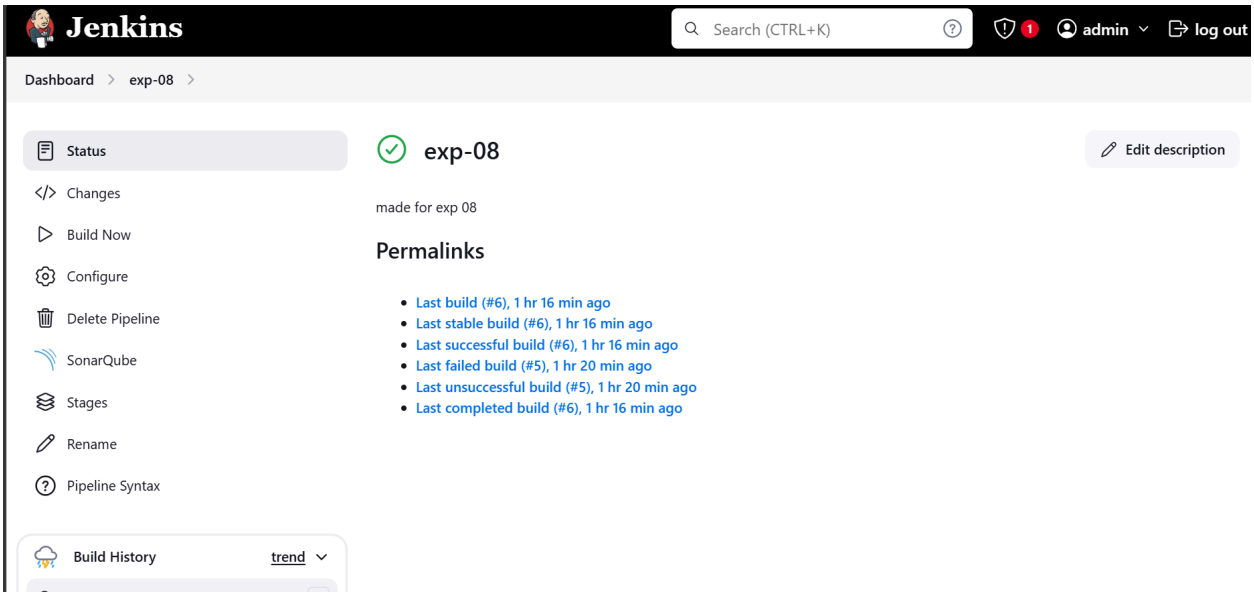
Name: Mohit Kerkar Div: D15C Roll No: 23



If you would notice carefully, i have used the withCredentials function within the pipeline script which makes the build avail the facility of using the credentials that i had set earlier

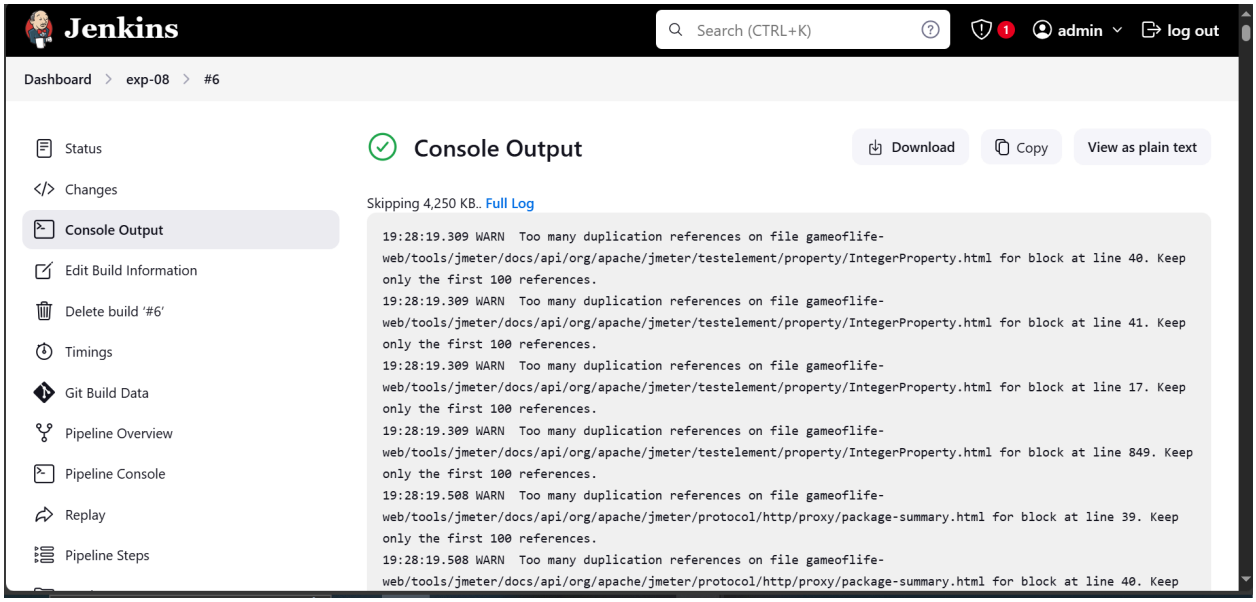
I have also added a line where i have mentioned the path of the sonar-scanner.bat file, since there are problems with system configurations with the installation of soanrqube

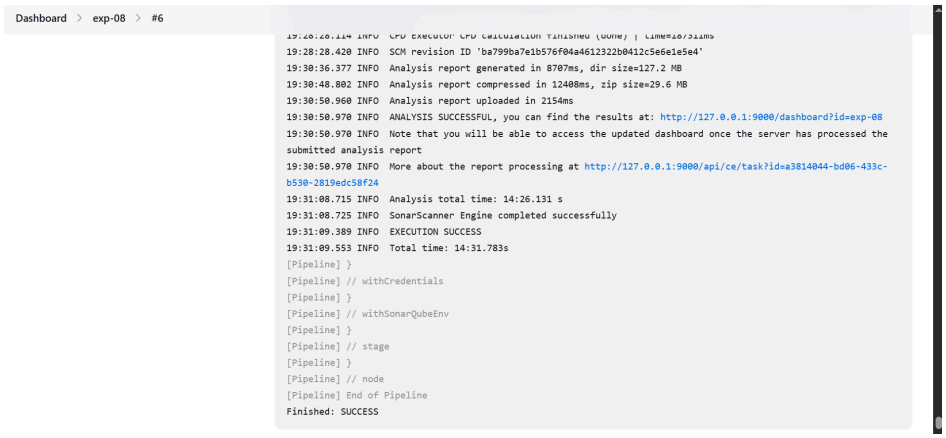
Now, save and build the pipeline



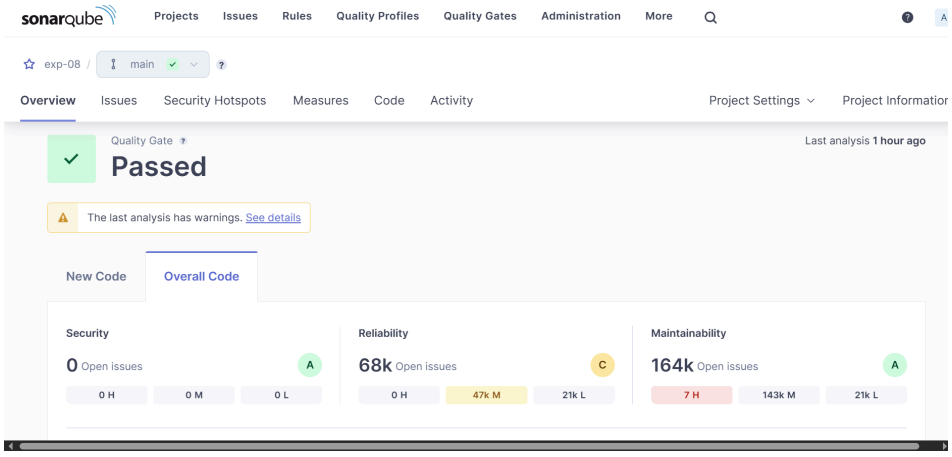
It generally takes 10-15 minutes for building this pipeline. We are supposed to stay patient and let the build process get completed

Name: Mohit Kerkar Div: D15C Roll No: 23





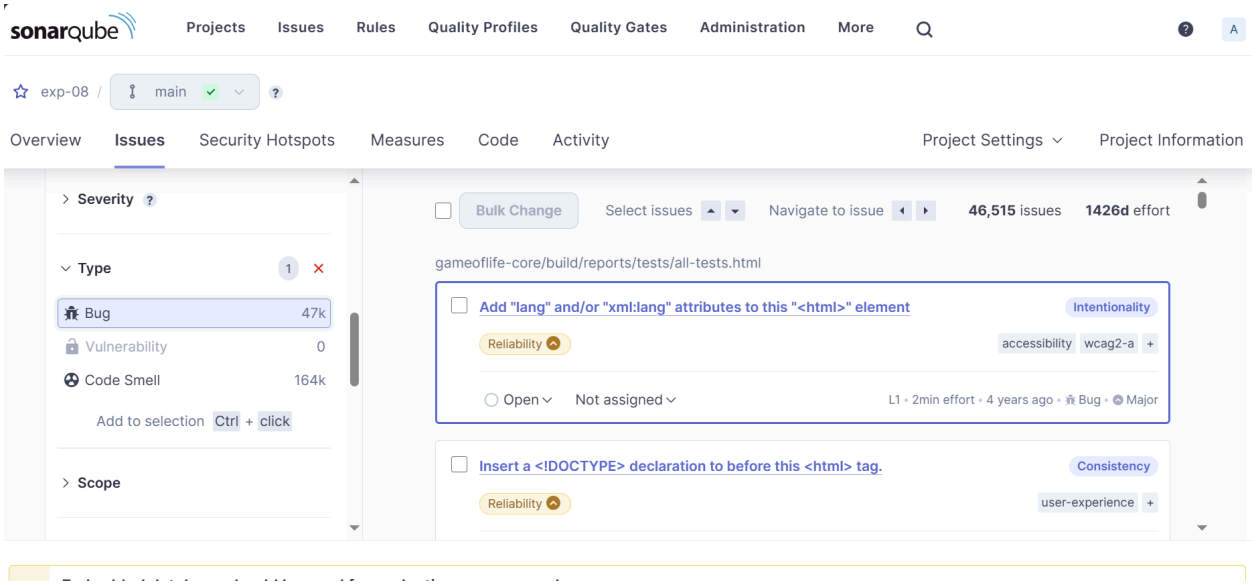
After a successful build, we are supposed to go back to our Sonarqube profile. We will see this type of a message on the screen, where it offers us multiple options to analyze our code



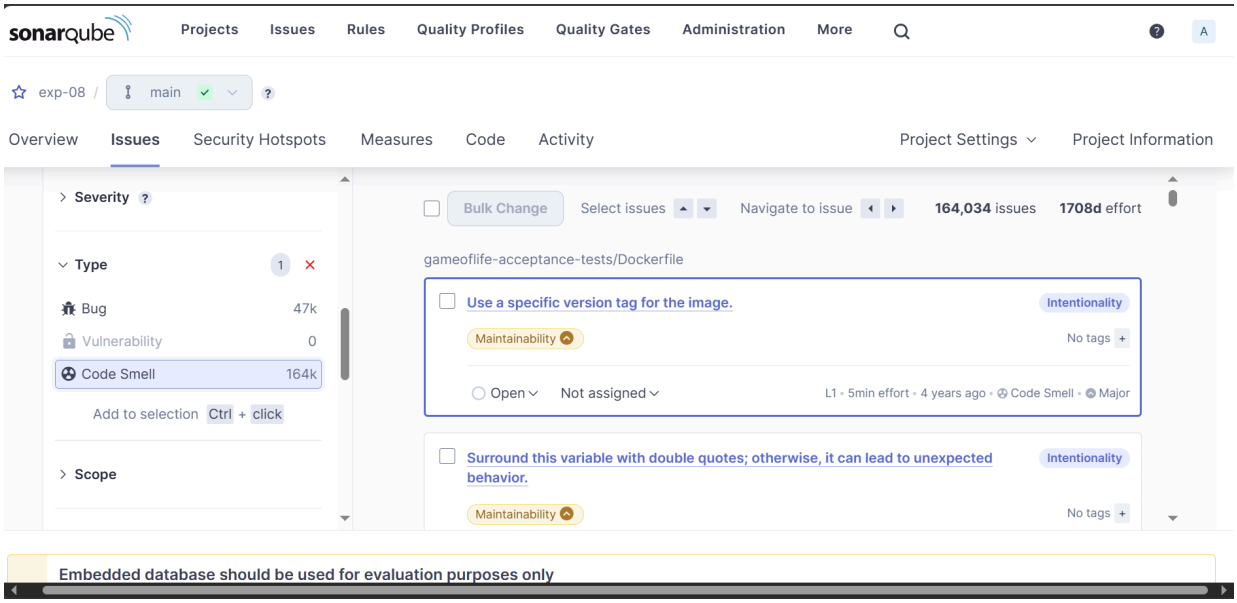
Within the issues section, let us explore different tabs

Name: Mohit Kerkar Div: D15C Roll No: 23

1. Bugs inside Type section

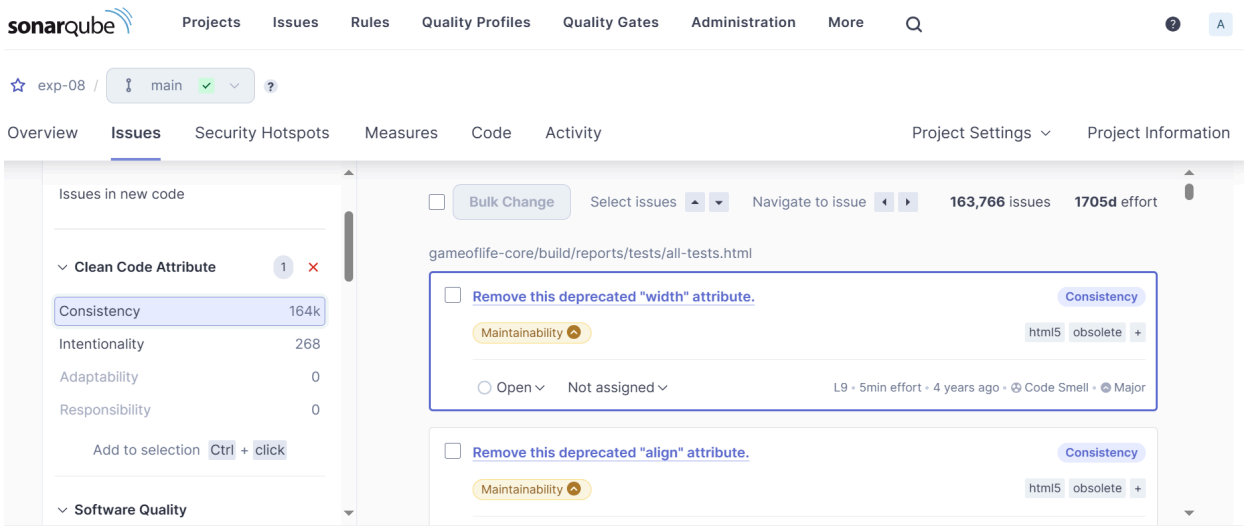


2. Code Smell inside the Type section

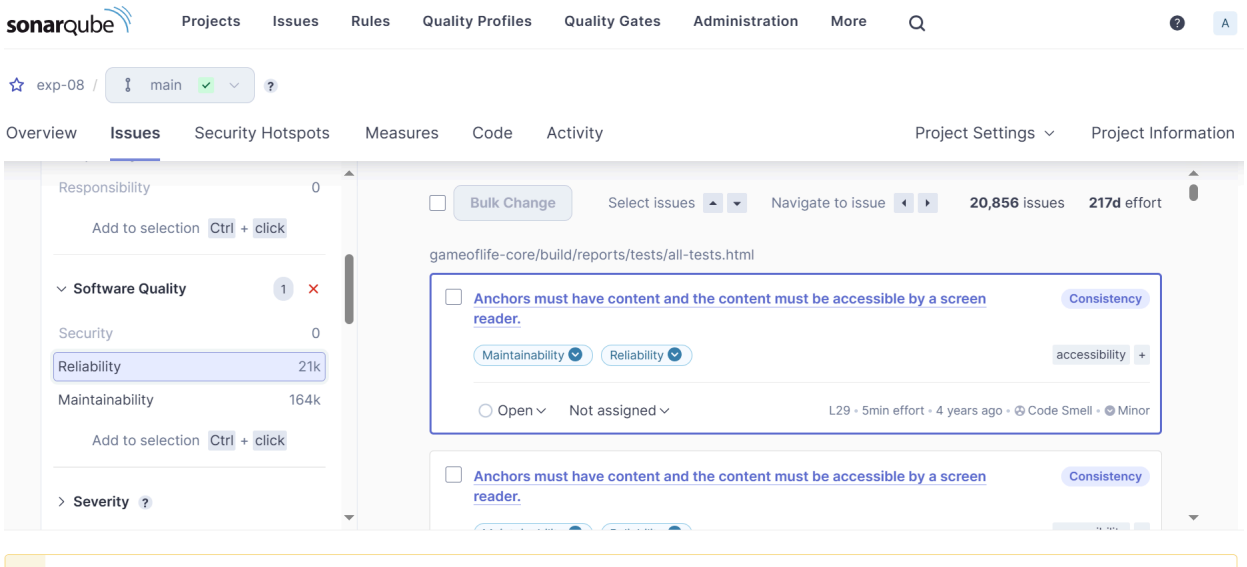


3. Consistency inside Clean Code Attribute

Name: Mohit Kerkar Div: D15C Roll No: 23

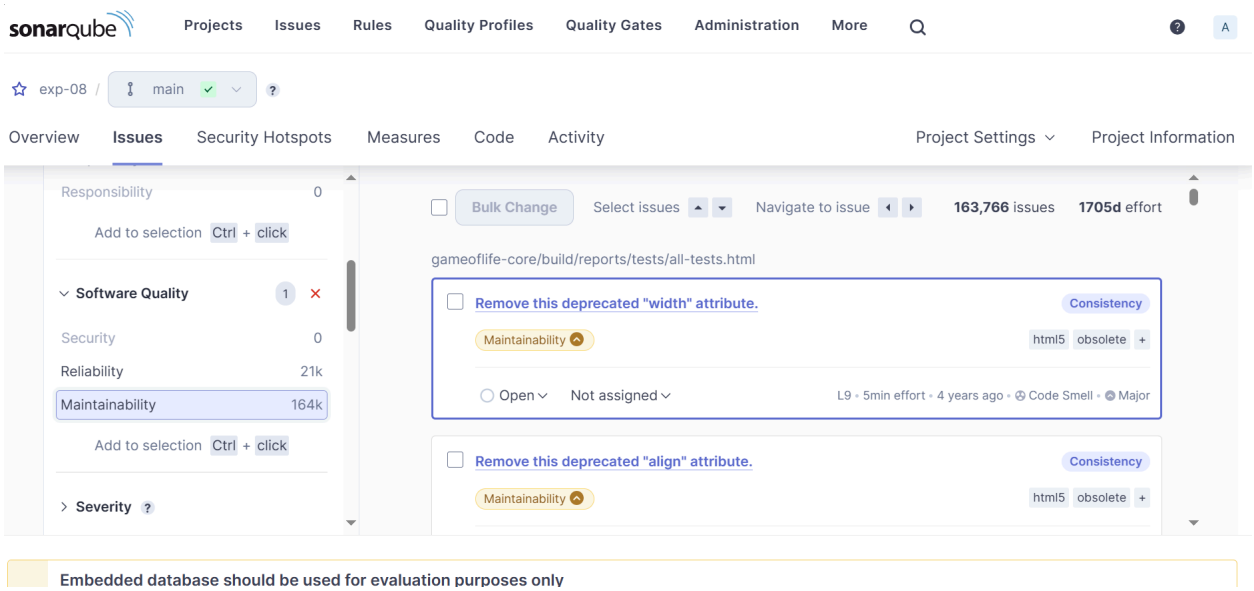


4. Reliability inside Software Quality



Name: Mohit Kerkar Div: D15C Roll No: 23

5. Maintainability



Conclusion:

In this experiment, we set up a Jenkins CI/CD pipeline integrated with SonarQube to automate static analysis on a sample application. Jenkins was configured to trigger builds and run SonarQube's analysis with every code change, detecting bugs, code smells, and security vulnerabilities. This pipeline provided continuous monitoring and ensured early detection of issues, improving code quality and security. The experiment showcased how integrating CI/CD pipelines with SonarQube enhances development efficiency and ensures better, more reliable software

Name: Mohit Kerkar Div: D15C Roll No: 23

Adv DevOps Exp 09

Aim: To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

Theory:

What is Nagios?

Nagios is an open-source software for continuous monitoring of systems, networks, and infrastructures. It runs plugins stored on a server that is connected with a host or another server on your network or the Internet. In case of any failure, Nagios alerts about the issues so that the technical team can perform the recovery process immediately.
Nagios is used for continuous monitoring of systems, applications, service and business processes in a DevOps culture.

Why We Need Nagios tool?

Here are the important reasons to use Nagios monitoring tool:

- Detects all types of network or server issues
- Helps you to find the root cause of the problem which allows you to get the permanent solution to the problem
- Active monitoring of your entire infrastructure and business processes
- Allows you to monitor and troubleshoot server performance issues
- Helps you to plan for infrastructure upgrades before outdated systems create failures
- You can maintain the security and availability of the service
- Automatically fix problems in a panic situation

Features of Nagios

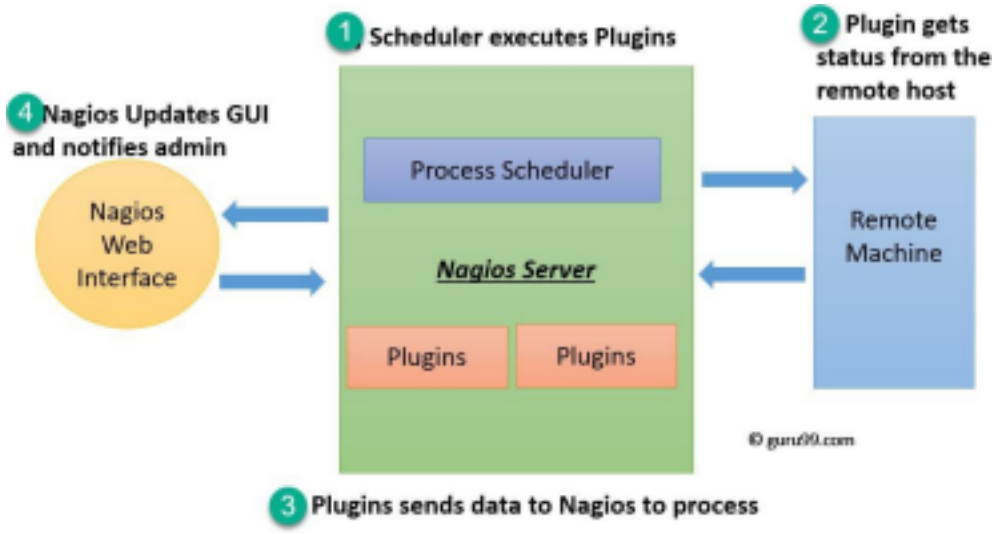
Following are the important features of Nagios monitoring tool:

- Relatively scalable, Manageable, and Secure
- Good log and database system
- Informative and attractive web interfaces
- Automatically send alerts if condition changes
- If the services are running fine, then there is no need to do check that host is an alive
- Helps you to detect network errors or server crashes
- You can troubleshoot the performance issues of the server.
- The issues, if any, can be fixed automatically as they are identified during the monitoring process
- You can monitor the entire business process and IT infrastructure with a single pass
- The product's architecture is easy to write new plugins in the language of your choice
- Nagios allows you to read its configuration from an entire directory which helps you to decide how to define individual files
- Utilizes topology to determine dependencies
- Monitor network services like HTTP, SMTP, HTTP, SNMP, FTP, SSH, POP, etc.
- Helps you to define network host hierarchy using parent hosts
- Ability to define event handlers that runs during service or host events for proactive

problem resolution
- Support for implementing redundant monitoring hosts

Nagios Architecture

Nagios is a client-server architecture. Usually, on a network, a Nagios server is running on a host, and plugins are running on all the remote hosts which should be monitored.

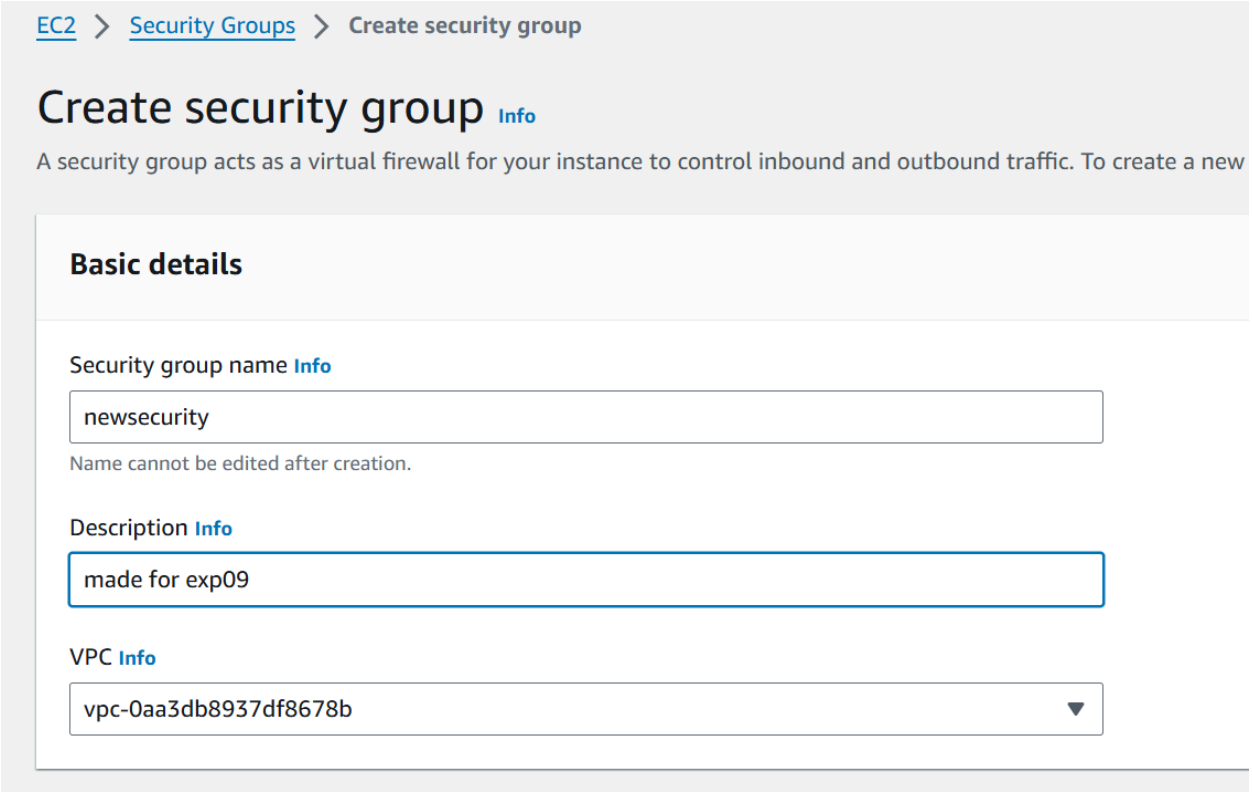


1. The scheduler is a component of the server part of Nagios. It sends a signal to execute the plugins at the remote host.
2. The plugin gets the status from the remote host
3. The plugin sends the data to the process scheduler
4. The process scheduler updates the GUI and notifications are sent to admins.

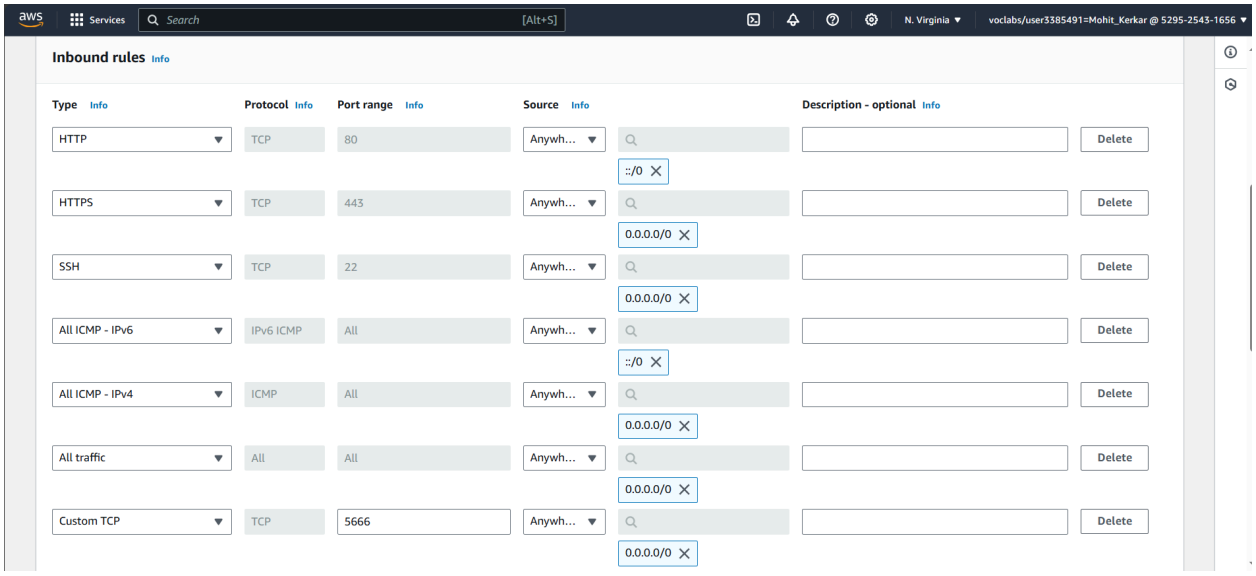
Name: Mohit Kerkar Div: D15C Roll No: 23

Step 1: Create a security group with the required configurations

I have created a new security group with a name ‘newsecurity’



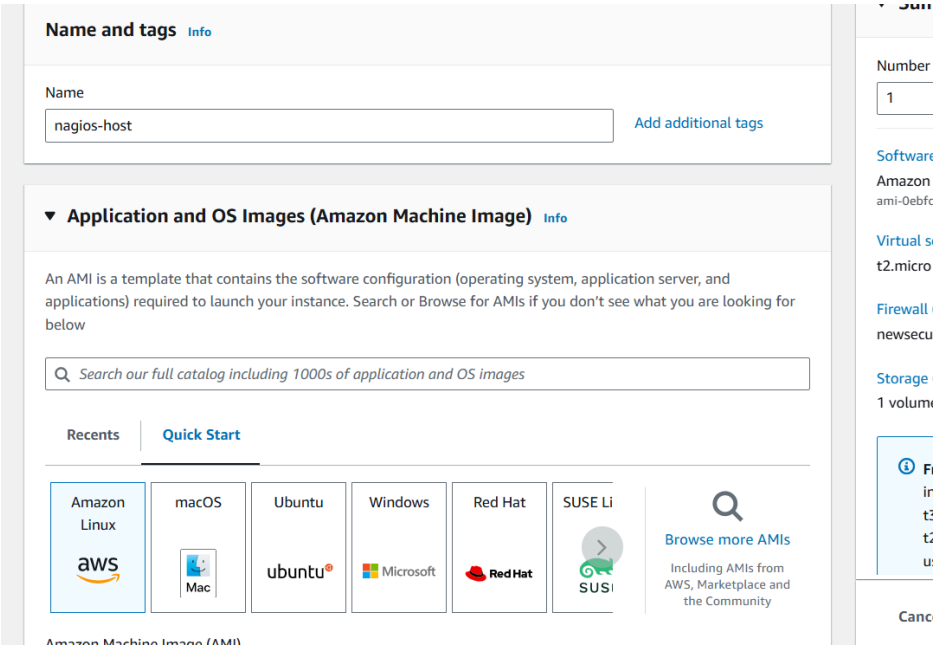
I have modified the INBOUND RULES as follows

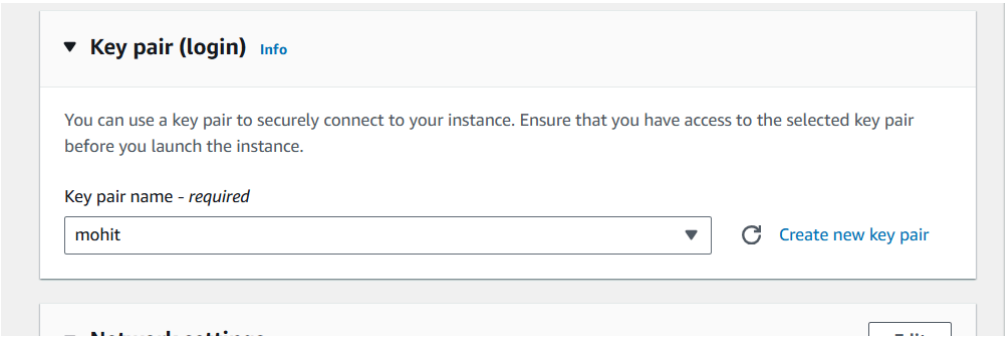


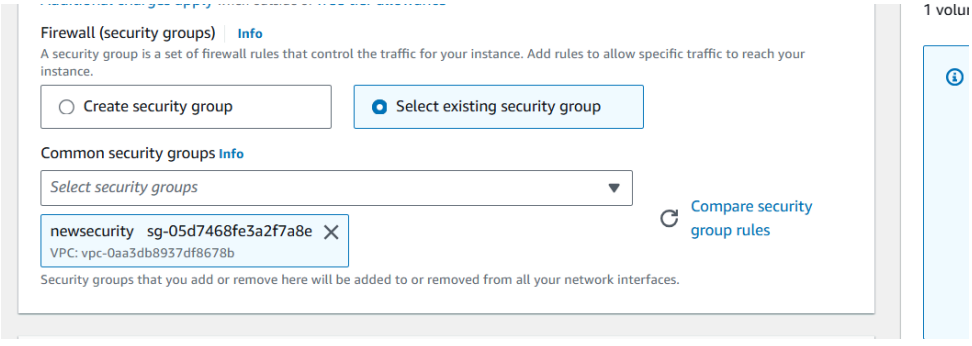
Name: Mohit Kerkar Div: D15C Roll No: 23

Step 2: Create ec2 instance

Name it as nagios-host. Select instance type as amazon-linux and choose the already created key pair and security group

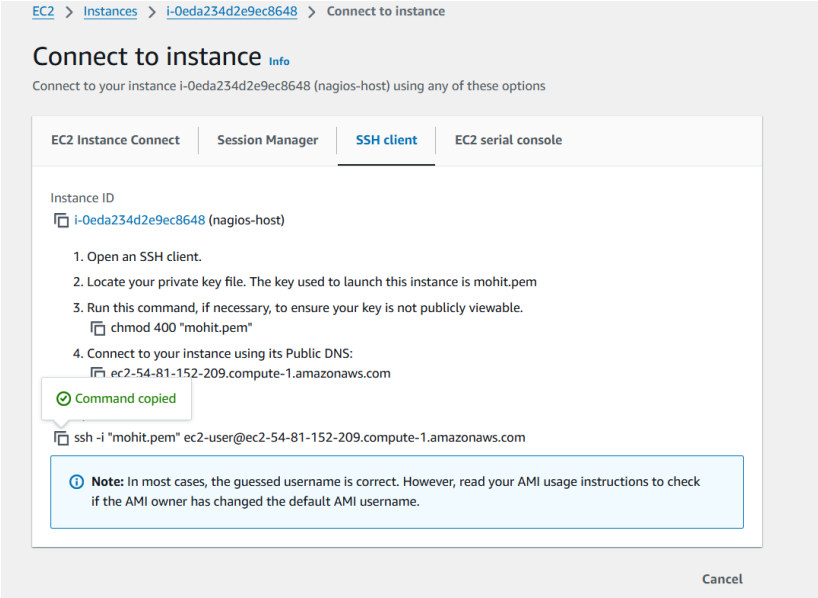




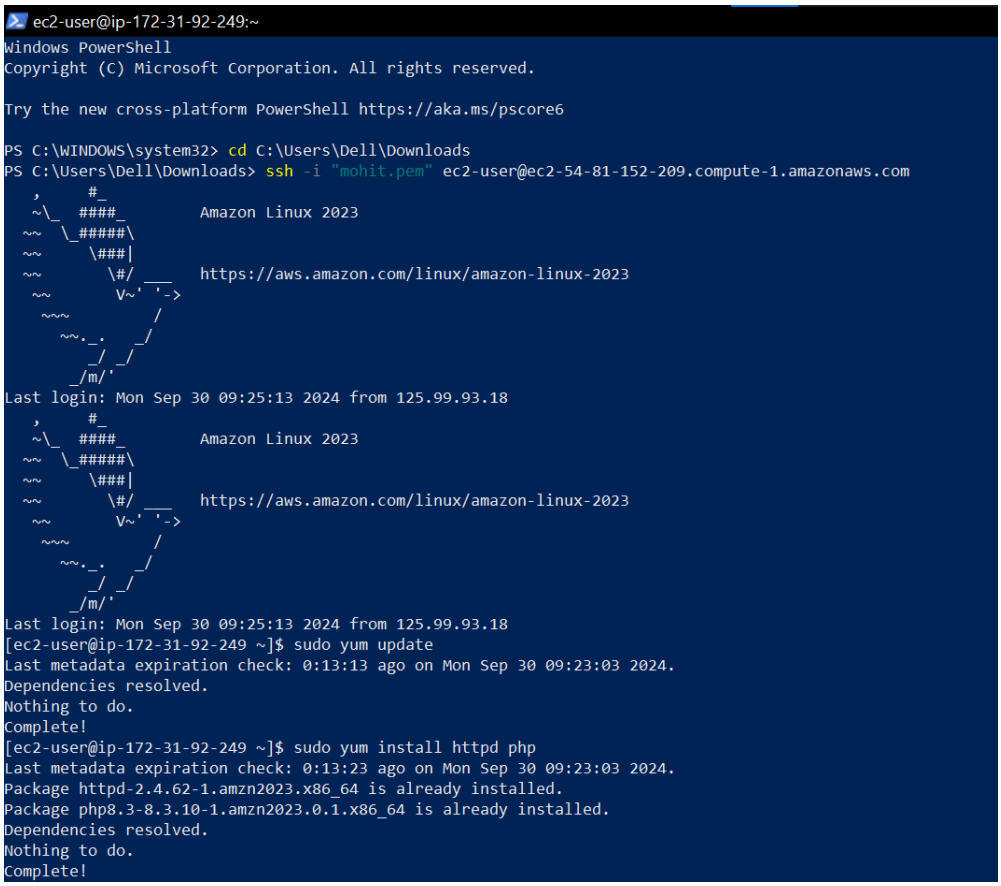


Name: Mohit Kerkar Div: D15C Roll No: 23

Copy the given ssh command, as we will require it for logging into our nagios-host instance from our windows powershell



Step 3: Open an administrative powershell and remotely login using the above mentioned ssh command

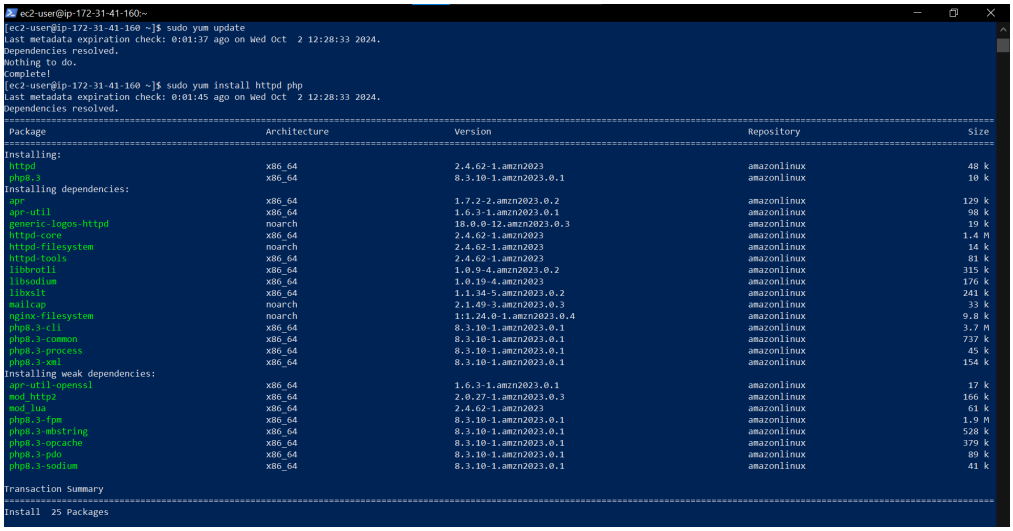


Name: Mohit Kerkar Div: D15C Roll No: 23

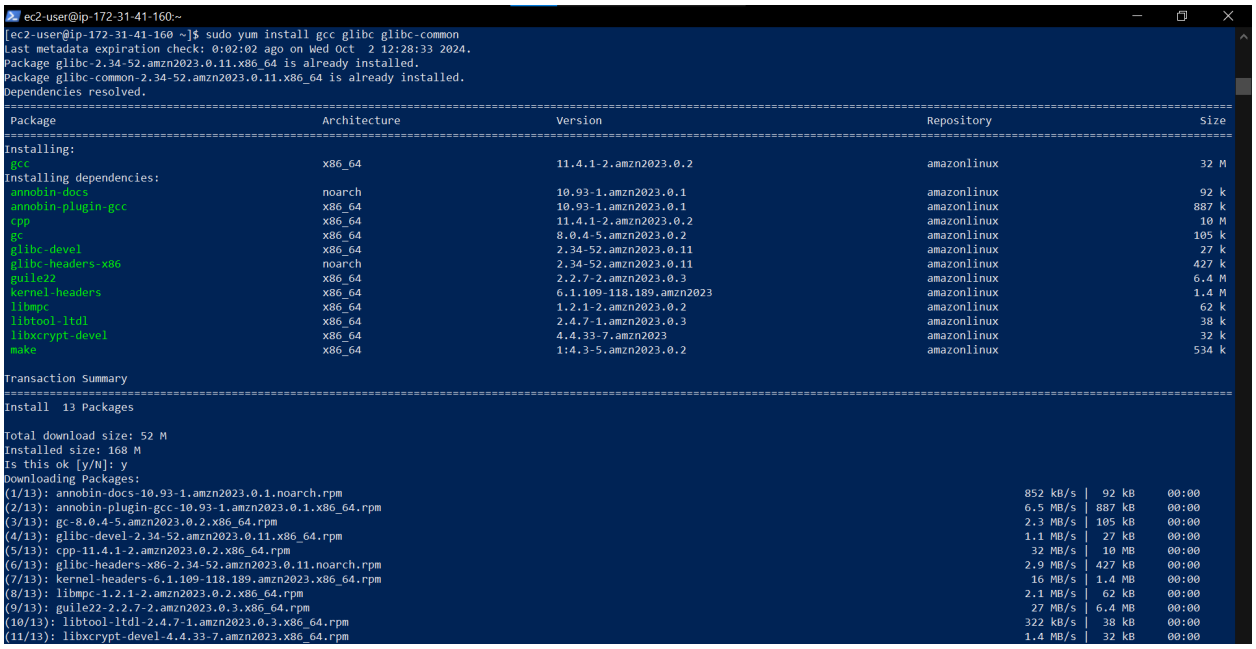
And then run these commands

sudo yum update

sudo yum install httpd php



sudo yum install gcc glibc glibc-common



sudo yum install gd gd-devel

```
ec2-user@ip-172-31-41-160:~
[ec2-user@ip-172-31-41-160 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:02:25 ago on Wed Oct  2 12:28:33 2024.
Dependencies resolved.
================================================================================
 Package                Architecture    Version                  Repository  Size
================================================================================
Installing:
 gd                     x86_64          2.3.3-5.amzn2023.0.3     amazonlinux 139 k
 gd-devel               x86_64          2.3.3-5.amzn2023.0.3     amazonlinux  38 k
Installing dependencies:
 brotli                 x86_64          1.0.9-4.amzn2023.0.2     amazonlinux 314 k
 brotli-devel           x86_64          1.0.9-4.amzn2023.0.2     amazonlinux  31 k
 bzip2-devel            x86_64          1.0.8-6.amzn2023.0.2     amazonlinux 214 k
 cairo                  x86_64          1.17.6-2.amzn2023.0.1    amazonlinux 684 k
 cmake-filesystem       x86_64          3.22.2-1.amzn2023.0.4    amazonlinux  16 k
 fontconfig             x86_64          2.13.94-2.amzn2023.0.2   amazonlinux 273 k
 fontconfig-devel       x86_64          2.13.94-2.amzn2023.0.2   amazonlinux 128 k
 fonts-filesystem       noarch          1:2.0.5-12.amzn2023.0.2  amazonlinux 9.5 k
 freetype               x86_64          2.13.2-5.amzn2023.0.1    amazonlinux 423 k
 freetype-devel         x86_64          2.13.2-5.amzn2023.0.1    amazonlinux 912 k
 glib2-devel            x86_64          2.74.7-689.amzn2023.0.2  amazonlinux 486 k
 google-noto-fonts-common noarch        20201206-2.amzn2023.0.2  amazonlinux  15 k
 google-noto-sans-vf-fonts noarch       20201206-2.amzn2023.0.2  amazonlinux 492 k
 graphite2              x86_64          1.3.14-7.amzn2023.0.2    amazonlinux  97 k
 graphite2-devel        x86_64          1.3.14-7.amzn2023.0.2    amazonlinux  21 k
 harfbuzz               x86_64          7.0.0-2.amzn2023.0.1     amazonlinux 868 k
 harfbuzz-devel         x86_64          7.0.0-2.amzn2023.0.1     amazonlinux 404 k
 harfbuzz-icu           x86_64          7.0.0-2.amzn2023.0.1     amazonlinux  18 k
 jbigkit-libs           x86_64          2.1-21.amzn2023.0.2      amazonlinux  54 k
 langpacks-core-font-en noarch          3.0-21.amzn2023.0.4      amazonlinux  10 k
 libICE                 x86_64          1.0.10-6.amzn2023.0.2    amazonlinux  71 k
 libSM                  x86_64          1.2.3-8.amzn2023.0.2     amazonlinux  42 k
 libX11                 x86_64          1.7.2-3.amzn2023.0.4     amazonlinux 657 k
 libX11-common          noarch          1.7.2-3.amzn2023.0.4     amazonlinux 152 k
 libX11-devel           x86_64          1.7.2-3.amzn2023.0.4     amazonlinux 939 k
 libX11-xcb             x86_64          1.7.2-3.amzn2023.0.4     amazonlinux  12 k
 libXau                 x86_64          1.0.9-6.amzn2023.0.2     amazonlinux  31 k
 libXau-devel           x86_64          1.0.9-6.amzn2023.0.2     amazonlinux  14 k
 libXext                x86_64          1.3.4-6.amzn2023.0.2     amazonlinux  41 k
 libXpm                 x86_64          3.5.15-2.amzn2023.0.3    amazonlinux  65 k
 libXpm-devel           x86_64          3.5.15-2.amzn2023.0.3    amazonlinux  59 k
 libXrender             x86_64          0.9.10-14.amzn2023.0.2   amazonlinux  28 k
 libXt                  x86_64          1.2.0-4.amzn2023.0.2     amazonlinux 181 k
 libblkid-devel         x86_64          2.37.4-1.amzn2023.0.4    amazonlinux  15 k
```

Create a new Nagios User with its password. You'll have to enter the password twice for confirmation.

sudo adduser -m nagios

sudo passwd nagios

```
[ec2-user@ip-172-31-41-160 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-41-160 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-41-160 ~]$
```

Create a new user group & create a new directory for Nagios downloads using the following commands

sudo groupadd nagcmd

sudo usermod -a -G nagcmd nagios

sudo usermod -a -G nagcmd apache

mkdir ~/downloads

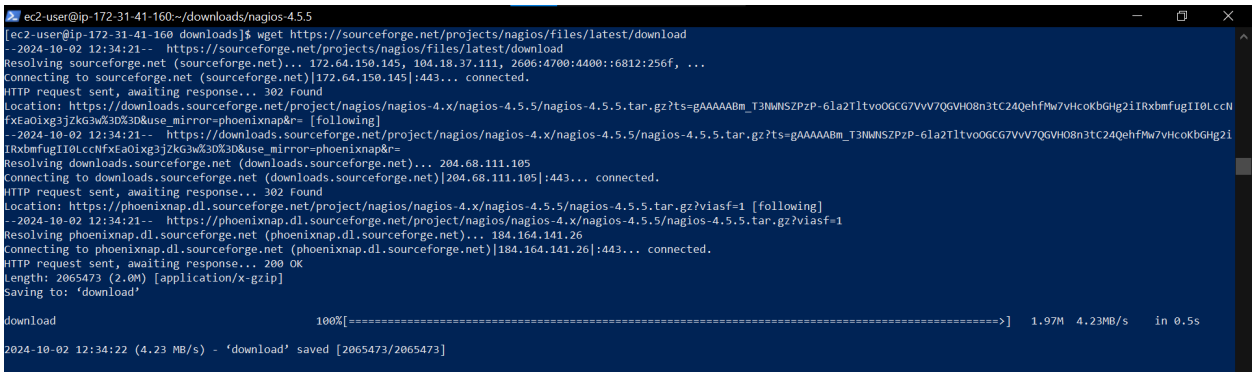
cd ~/downloads

Use **wget** to download the source zip files.

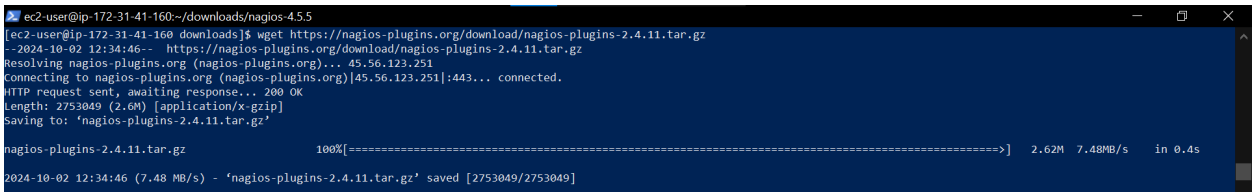
In this step, we are downloading, the latest version of nagios and the necessary plugins required to carry out the tasks of setting up a nagios server

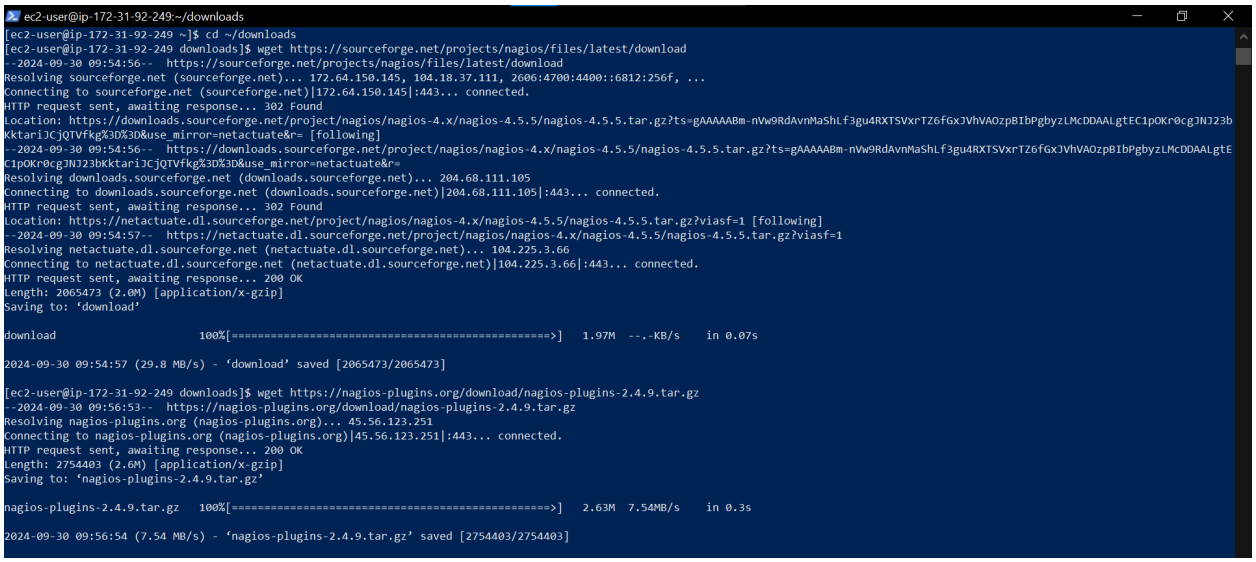
Name: Mohit Kerkar Div: D15C Roll No: 23

wget https://sourceforge.net/projects/nagios/files/latest/download



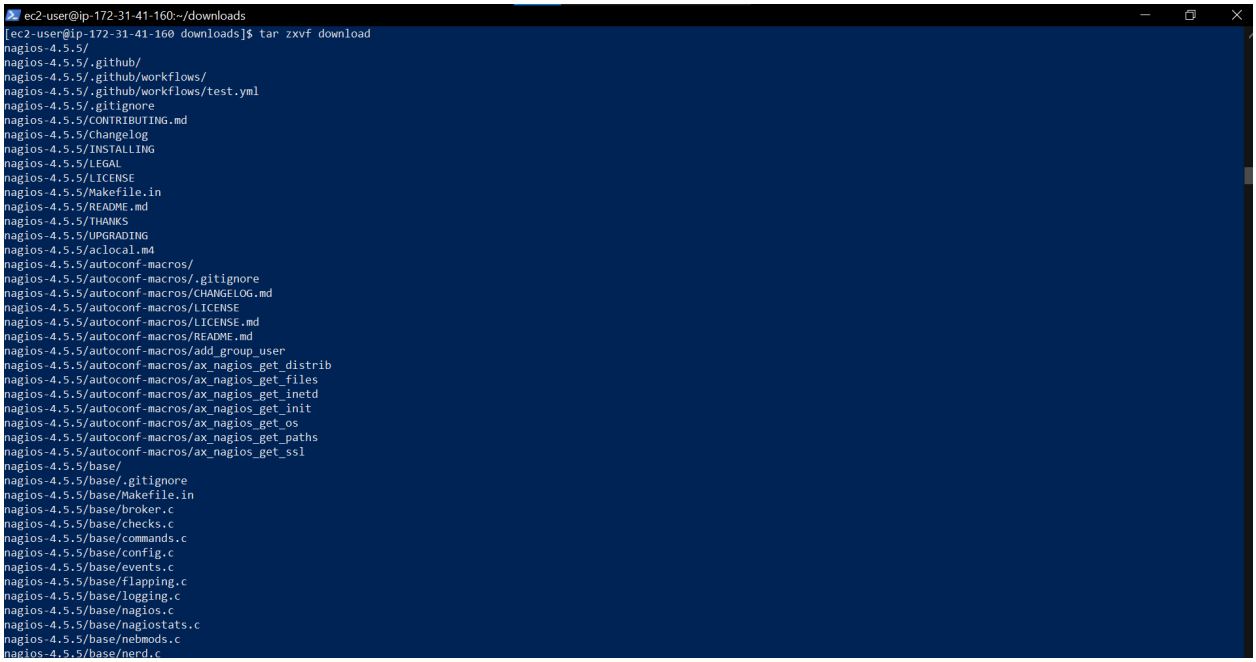
wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz





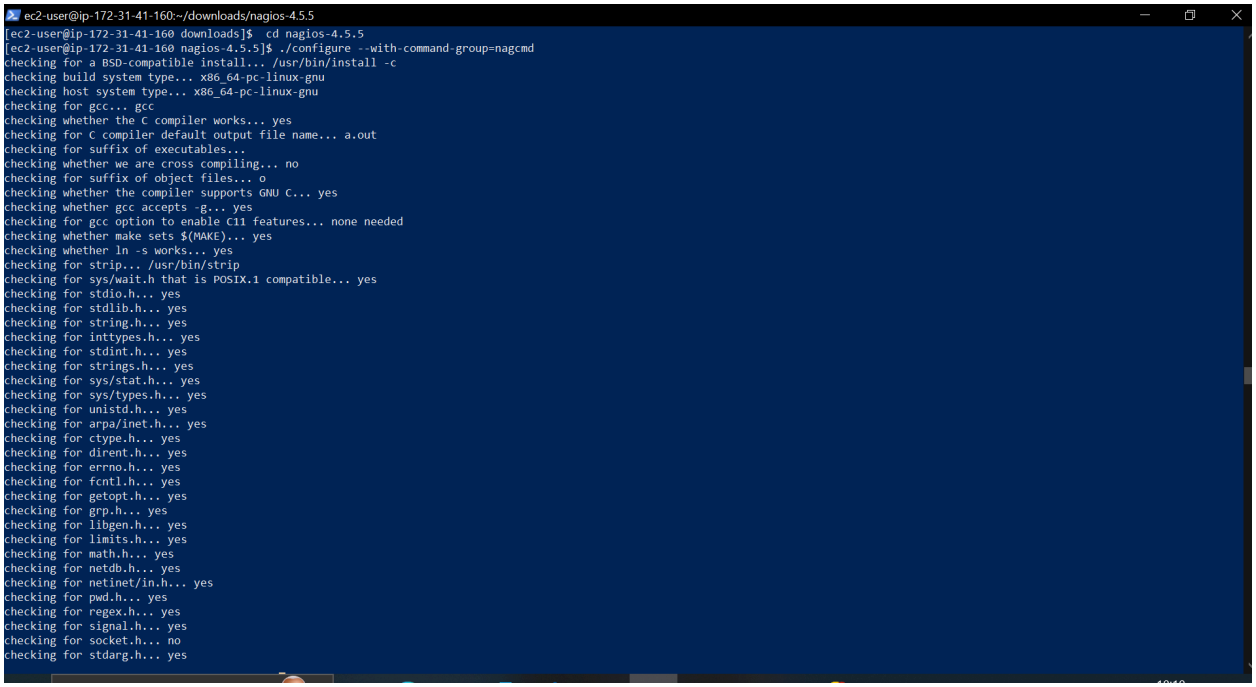
Now, we run the next command in the following manner
tar zxvf <nagios-4.5.5 version> (for me it has gotten saved as 'download')
So i wrote **tar zxvf download**

Name: Mohit Kerkar Div: D15C Roll No: 23



After which we are supposed to **change our directory** over there
For eg. **cd nagios-4.5.5**… depending on the version that we have downloaded

Next, Run this command (make sure that you are working inside nagios-4.x.x directory)
./configure --with-command-group=nagcmd



```
checking for strstr... yes
checking for strtoul... yes
checking for unsetenv... yes
checking for type of socket size... size_t
checking for Kerberos include files... configure: WARNING: could not find include files
checking for pkg-config... pkg-config
checking for SSL headers... configure: error: Cannot find ssl headers
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

After running this command, we get an **error related to ssl header being absent**

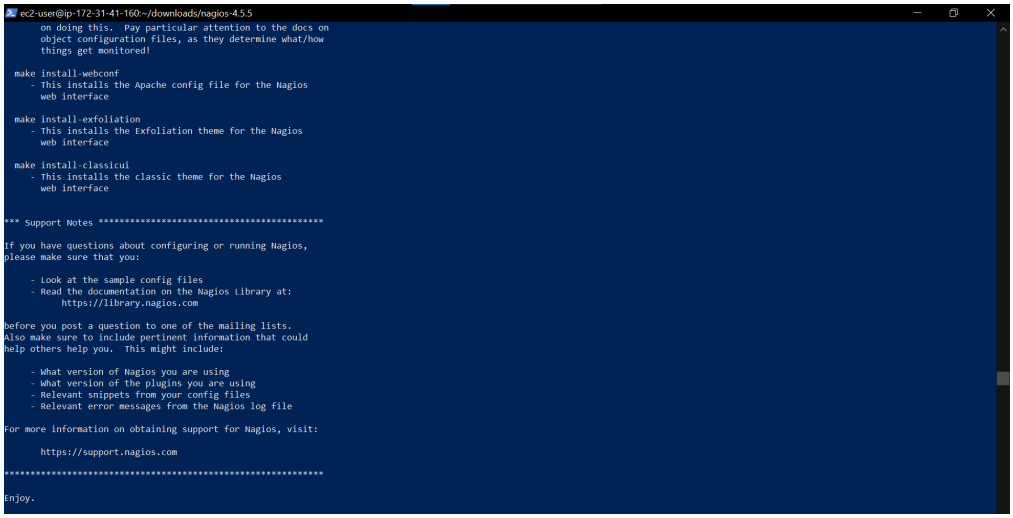
For that purpose, we are to run the following command.
sudo yum install openssl-devel (for ssl header)

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo yum install openssl-devel
Last metadata expiration check: 0:12:11 ago on Wed Oct  2 12:28:33 2024.
Dependencies resolved.
==========================================================================================================================
 Package                    Architecture           Version                                Repository                 Size
==========================================================================================================================
Installing:
 openssl-devel              x86_64                 1:3.0.8-1.amzn2023.0.14                amazonlinux               3.0 M

Transaction Summary
==========================================================================================================================
Install  1 Package

Total download size: 3.0 M
Installed size: 4.7 M
Is this ok [y/N]: y
Downloading Packages:
openssl-devel-3.0.8-1.amzn2023.0.14.x86_64.rpm                                             26 MB/s | 3.0 MB     00:00
--------------------------------------------------------------------------------------------------------------------------
Total                                                                                      17 MB/s | 3.0 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                  1/1
  Installing       : openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64                                                     1/1
  Running scriptlet: openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64                                                     1/1
  Verifying        : openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64                                                     1/1

Installed:
  openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64

Complete!
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

Now, Re-run **./configure --with-command-group=nagcmd**
After this, run **make all** command

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o ./nagios.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o broker.o broker.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o nebmods.o nebmods.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o ../common/shared.o ../common/shared.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o query-handler.o query-handler.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o workers.o workers.c
In function ‘get_wproc_list’,
    inlined from ‘get_worker’ at workers.c:277:12:
workers.c:253:17: warning: ‘%s’ directive argument is null [-Wformat-overflow=]
  253 |                 log_debug_info(DEBUGL_CHECKS, 1, "Found specialized worker(s) for '%s'", (slash && *slash != '/') ? slash : cmd_name);
      |                 ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o checks.o checks.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o config.o config.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o commands.o commands.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o events.o events.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o flapping.o flapping.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o logging.o logging.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o macros-base.o ../common/macros.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o netutils.o netutils.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o notifications.o notifications.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o sehandlers.o sehandlers.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o utils.o utils.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o retention-base.o ./sretention.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xretention-base.o ../xdata/xrddefault.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o comments-base.o ../common/comments.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xcomments-base.o ../xdata/xcddefault.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o objects-base.o ../common/objects.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xobjects-base.o ../xdata/xodtemplate.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o statusdata-base.o ../common/statusdata.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xstatusdata-base.o ../xdata/xsddefault.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o perfdata-base.o ./perfdata.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xperfdata-base.o ../xdata/xpddefault.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o downtime-base.o ../common/downtime.c
make -C ../lib
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/lib'
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c squeue.c -o squeue.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c kvvec.c -o kvvec.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c iocache.c -o iocache.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c iobroker.c -o iobroker.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c bitmap.c -o bitmap.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c dkhash.c -o dkhash.o
```

Name: Mohit Kerkar Div: D15C Roll No: 23



Run the following set of commands to ensure that
sudo make install

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
        /usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
cd ./html && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/html'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/media
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/stylesheets
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/contexthelp
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs/images
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/js
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/images
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/images/logos
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/includes
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/ssi
/usr/bin/install -c -m 664 -o nagios -g nagios ./robots.txt /usr/local/nagios/share
/usr/bin/install -c -m 664 -o nagios -g nagios ./jsonquery.html /usr/local/nagios/share
rm -f /usr/local/nagios/share/index.html
rm -f /usr/local/nagios/share/main.html
rm -f /usr/local/nagios/share/side.html
rm -f /usr/local/nagios/share/map.html
rm -f /usr/local/nagios/share/rss-*
rm -f /usr/local/nagios/share/graph-header.html
rm -f /usr/local/nagios/share/histogram.html
rm -f /usr/local/nagios/share/histogram-form.html
rm -f /usr/local/nagios/share/histogram-graph.html
rm -f /usr/local/nagios/share/histogram-links.html
rm -f /usr/local/nagios/share/infobox.html
rm -f /usr/local/nagios/share/map.php
```

sudo make install-init

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install-init
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
```

sudo make install-config

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install-config
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg

*** Config files installed ***

Remember, these are *SAMPLE* config files.  You'll need to read
the documentation for more information on how to actually define
services, hosts, etc. to fit your particular needs.
```

sudo make install-webconf

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

*** Nagios/Apache conf file installed ***

[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

Next, we are supposed to create a nagiosadmin account for nagios login along with password. Specify the password twice.

sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

Restart Apache

sudo service httpd restart

Go back to the downloads folder and unzip the plugins zip file.

cd ~/downloads

tar zxvf nagios-plugins-2.4.11.tar.gz

```
ec2-user@ip-172-31-41-160:~/downloads
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ cd ~/downloads
[ec2-user@ip-172-31-41-160 downloads]$ tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
nagios-plugins-2.4.11/build-aux/depcomp
nagios-plugins-2.4.11/build-aux/snippet/
nagios-plugins-2.4.11/build-aux/snippet/_Noreturn.h
nagios-plugins-2.4.11/build-aux/snippet/arg-nonnull.h
nagios-plugins-2.4.11/build-aux/snippet/c++defs.h
nagios-plugins-2.4.11/build-aux/snippet/warn-on-use.h
nagios-plugins-2.4.11/build-aux/test-driver
```

Compile and install plugins
cd nagios-plugins-2.4.11
./configure --with-nagios-user=nagios --with-nagios-group=nagios

Run the following command:
sudo chkconfig --add nagios
On running the above command

```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
error reading information on service nagios: No such file or directory
```

If this is the output that one is getting, then it means that the init script is missing…
We can check this by running ls /etc/init.d/

```
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.9]$ ls /etc/init.d/
README  functions
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.9]$
```

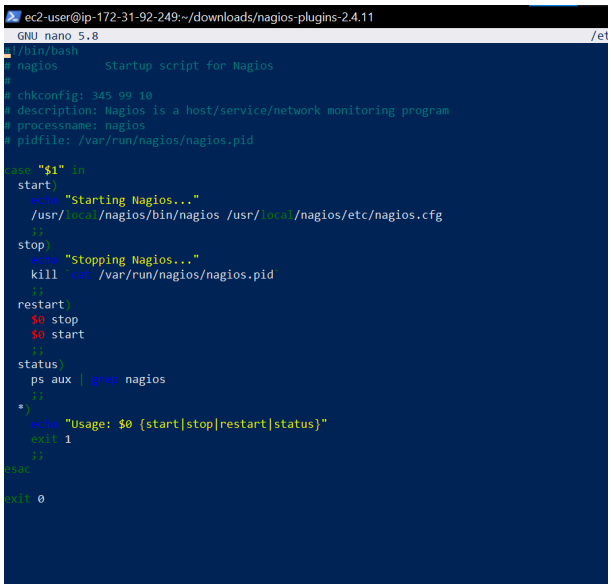
With ls command, we must see a file named nagios, which i was not able to see
If the Init Script is Missing i.e If you don't see the nagios script in /etc/init.d/, you can create it manually. Here's how:

Run the following command:
sudo nano /etc/init.d/nagios

Within this file, paste the following script

Name: Mohit Kerkar Div: D15C Roll No: 23

```
#!/bin/bash
# nagios       Startup script for Nagios
#
# chkconfig: 345 99 10
# description: Nagios is a host/service/network monitoring program
# processname: nagios
# pidfile: /var/run/nagios/nagios.pid
case "$1" in
  start)
    echo "Starting Nagios..."
    /usr/local/nagios/bin/nagios /usr/local/nagios/etc/nagios.cfg
    ;;
  stop)
    echo "Stopping Nagios..."
    kill `cat /var/run/nagios/nagios.pid`
    ;;
  restart)
    $0 stop
    $0 start
    ;;
  status)
    ps aux | grep nagios
    ;;
  *)
    echo "Usage: $0 {start|stop|restart|status}"
    exit 1
    ;;
esac
exit 0
```



Name: Mohit Kerkar Div: D15C Roll No: 23

Make the Script Executable: After saving the file, run the following command to make it executable:
sudo chmod +x /etc/init.d/nagios

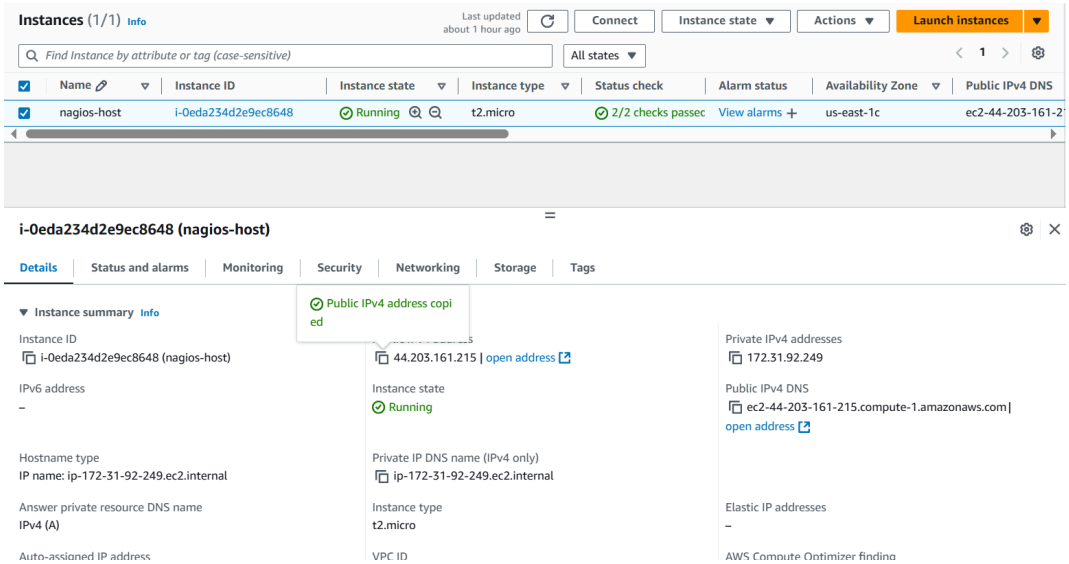
Run **sudo chkconfig --add nagios** again
And then run **sudo chkconfig nagios on**

```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo nano /etc/init.d/nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chmod +x /etc/init.d/nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig nagios on
Note: Forwarding request to 'systemctl enable nagios.service'.
Synchronizing state of nagios.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nagios
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$
```

sudo service nagios start

```
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.11]$ sudo service nagios start
Starting Nagios...

Nagios Core 4.5.5
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2024-09-17
License: GPL

Website: https://www.nagios.org
Nagios 4.5.5 starting... (PID=72261)
Local time is Tue Oct 01 20:59:58 UTC 2024
wproc: Successfully registered manager as @wproc with query handler
wproc: Registry request: name=Core Worker 72265;pid=72265
wproc: Registry request: name=Core Worker 72264;pid=72264
wproc: Registry request: name=Core Worker 72263;pid=72263
wproc: Registry request: name=Core Worker 72262;pid=72262
Successfully launched command file worker with pid 72266
wproc: NOTIFY job 4 from worker Core Worker 72262 is a non-check helper but exited with return code 127
wproc:   host=localhost; service=Swap Usage; contact=nagiosadmin
wproc:   early_timeout=0; exited_ok=1; wait_status=32512; error_code=0;
wproc:   stderr line 01: /bin/sh: line 1: /bin/mail: No such file or directory
wproc:   stderr line 02: /usr/bin/printf: write error: Broken pipe
```

Get your public IPv4 address from your instance. We will require it for connecting to our nginx server



Name: Mohit Kerkar Div: D15C Roll No: 23

Browse for this url: http://<your_public_ip_address>/nagios
The browser may ask you for your nagios credentials which set in the earlier steps
The username is nagiosadmin and enter the password that you set earlier

Conclusion:

In this experiment, we successfully installed and configured Nagios Core on an Amazon Linux EC2 instance, showcasing its role in continuous monitoring within a DevOps environment. We learned about user management and service configuration, emphasizing Nagios's ability to monitor systems and networks effectively. This experience laid the groundwork for enhancing infrastructure reliability and integrating advanced monitoring strategies in future projects.

Name: Mohit Kerkar Div: D15C Roll No: 23

Adv DevOps Exp 10

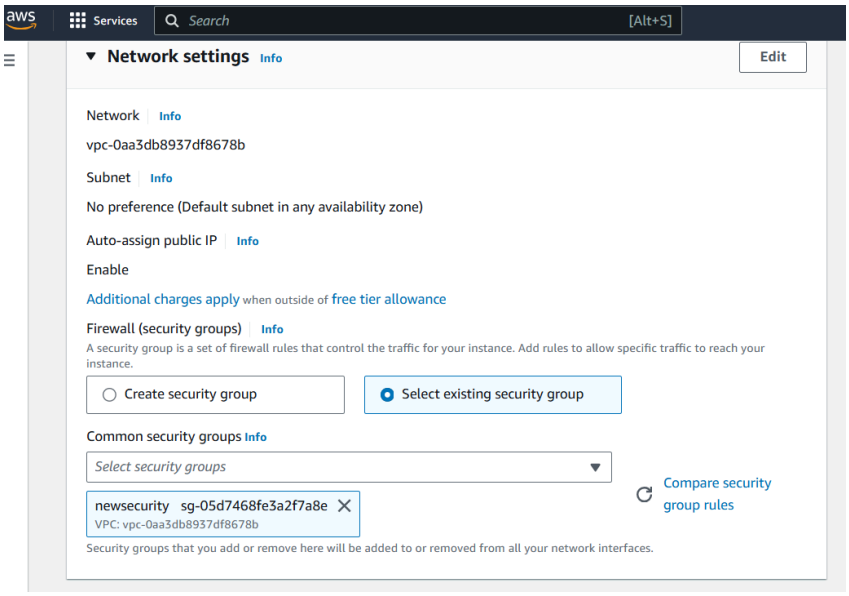
Aim: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

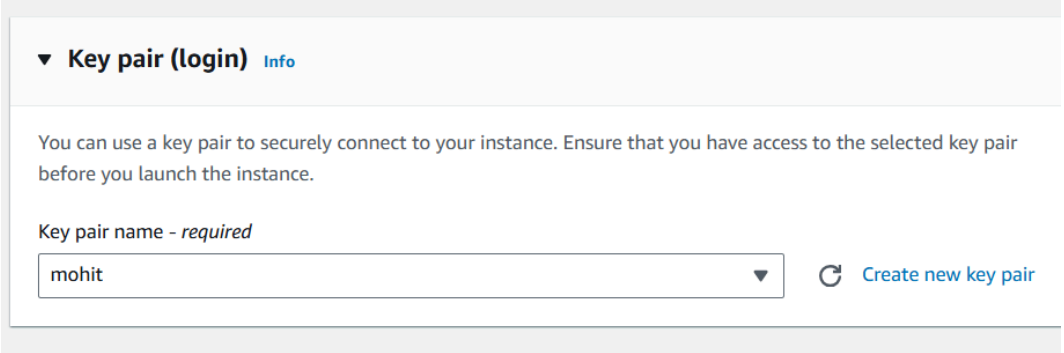
Monitoring Using Nagios:

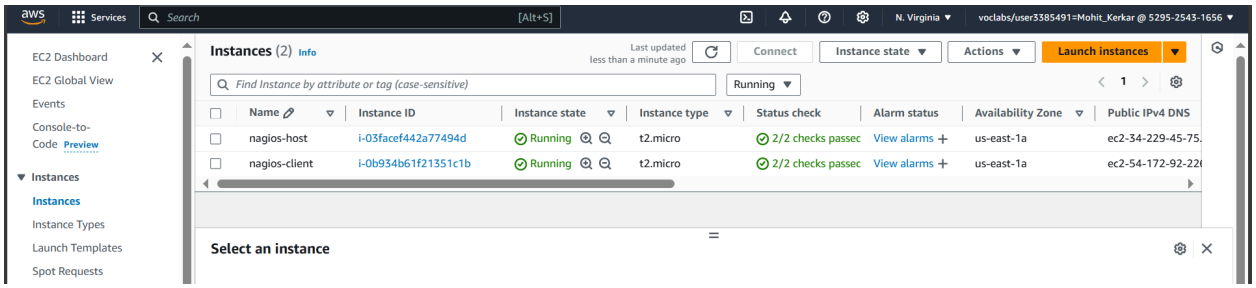
Step 1: To Confirm Nagios is running on the server side Perform the following command on your Amazon Linux Machine (Nagios-host).
Run this command **sudo systemctl status**

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-plugins-2.4.11
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo systemctl status
● ip-172-31-41-160.ec2.internal
    State: running
    Units: 296 loaded (incl. loaded aliases)
     Jobs: 0 queued
   Failed: 0 units
    Since: Wed 2024-10-02 12:28:05 UTC; 33min ago
  systemd: 252.23-2.amzn2023
  CGroup: /
          ├─init.scope
          │ └─1 /usr/lib/systemd/systemd --switched-root --system --deserialize=32
          ├─system.slice
          │ ├─acpid.service
          │ │ ├─1938 /usr/bin/systemd-inhibit --what=handle-suspend-key:handle-hibernate-key --who=noah "--why=acpid instead" --mode=block /usr/sbin/acpid -f
          │ │ └─2059 /usr/sbin/acpid -f
          │ ├─amazon-ssm-agent.service
          │ │ └─2141 /usr/bin/amazon-ssm-agent
          │ ├─atd.service
          │ │ └─2152 /usr/sbin/atd -f
          │ ├─auditd.service
          │ │ └─1768 /sbin/auditd
          │ ├─chronyd.service
          │ │ └─2175 /usr/sbin/chronyd -F 2
          │ ├─dbus-broker.service
          │ │ ├─1946 /usr/bin/dbus-broker-launch --scope system --audit
          │ │ └─1954 dbus-broker --log 4 --controller 9 --machine-id ec2e4d759a3e2f6fe850b14e4cdacabe --max-bytes 536870912 --max-fds 4096 --max-matches 16384 --audit
          │ ├─gssproxy.service
          │ │ └─1959 /usr/sbin/gssproxy -D
          │ ├─httpd.service
          │ │ ├─49553 /usr/sbin/httpd -DFOREGROUND
          │ │ ├─49555 /usr/sbin/httpd -DFOREGROUND
          │ │ ├─49556 /usr/sbin/httpd -DFOREGROUND
          │ │ ├─49557 /usr/sbin/httpd -DFOREGROUND
          │ │ ├─49558 /usr/sbin/httpd -DFOREGROUND
          │ │ └─62800 /usr/sbin/httpd -DFOREGROUND
          │ ├─libstoragemgmt.service
          │ │ └─1940 /usr/bin/lsmd -d
```

Step 2: Before we begin,
To monitor a Linux machine, create an **Ubuntu 20.04 server** EC2 Instance in AWS. Provide it with the **same security group** as the Nagios Host and name it ‘nagios-client’ alongside the host.







Step 3: TO BE DONE IN THE Nagios-host TERMINAL
In the nagios-host terminal, run this command
ps -ef | grep nagios

```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ ps -ef | grep nagios
ec2-user   63115    2315  0 13:03 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$
```

To become a root user, run **'sudo su'** and make two directories using the following commands. If one is running these commands in windows powershell, make sure that he/she copies it line by line as powershell might make an error while interpreting multiple lines
mkdir /usr/local/nagios/etc/objects/monitorhosts
mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts

```
[ec2-user@ip-172-31-92-249 ~]$ sudo su
[root@ip-172-31-92-249 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts
[root@ip-172-31-92-249 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-92-249 ec2-user]#
```

Copy the sample localhost.cfg file to linuxhost folder. Use the following mentioned command to achieve it
cp /usr/local/nagios/etc/objects/localhost.cfg
/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

Open linuxserver.cfg using nano and make the following changes. This is a conf type file in which we will have to modify the configurations in way which will help us specify the hosts and clients to be monitored

nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

Changes to be made:

1. Change the hostname to linux-server (EVERYWHERE ON THE FILE)
2. Change address to the public IP address of your LINUX CLIENT.
3. Change hostgroup_name under hostgroup to linux-servers1

```
# HOST DEFINITION
#
###############################################################################

# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.
    host_name               linux-server
    alias                   localhost
    address                 54.172.92.226
}



###############################################################################
```

```
# Define an optional hostgroup for Linux machines

define hostgroup {

    hostgroup_name          linux-servers1          ; The name of the hostgroup
    alias                   Linux Servers           ; Long name of the group
    members                 localhost               ; Comma separated list of hosts that belong to this group
}
```

IMP: Everywhere else on the file, change the hostname to linux-server instead of localhost.

Open the Nagios Config file and add the following line

nano /usr/local/nagios/etc/nagios.cfg

Add the following line in the file and save

cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/

```
# OBJECT CONFIGURATION FILE(S)
# These are the object configuration files in which you define hosts,
# host groups, contacts, contact groups, services, etc.
# You can split your object definitions across several config files
# if you wish (as shown below), or keep them all in a single config file.

# You can specify individual object config files as shown below:
cfg_file=/usr/local/nagios/etc/objects/commands.cfg
cfg_file=/usr/local/nagios/etc/objects/contacts.cfg
cfg_file=/usr/local/nagios/etc/objects/timeperiods.cfg
cfg_file=/usr/local/nagios/etc/objects/templates.cfg

# Definitions for monitoring the local (Linux) host
cfg_file=/usr/local/nagios/etc/objects/localhost.cfg
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
# Definitions for monitoring a Windows machine
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg
```

Verify the configuration files by running the following command

/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

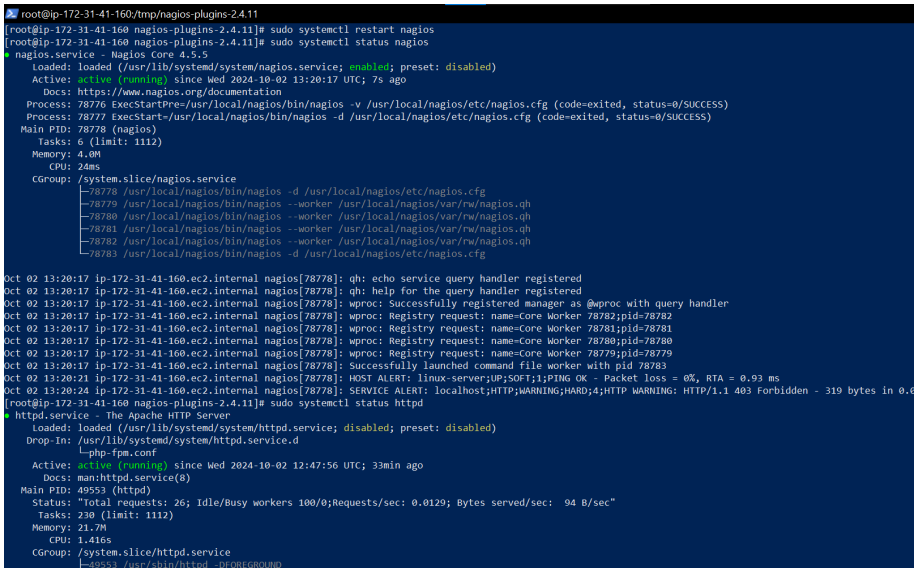
```
[root@ip-172-31-41-160 nagios-plugins-2.4.11]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.5.5
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2024-09-17
License: GPL

Website: https://www.nagios.org
Reading configuration data...
   Read main config file okay...
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
        Checked 16 services.
        Checked 2 hosts.
        Checked 2 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 2 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-41-160 nagios-plugins-2.4.11]#
```

You are good to go if there are no errors.

Restart the nagios service

service nagios restart

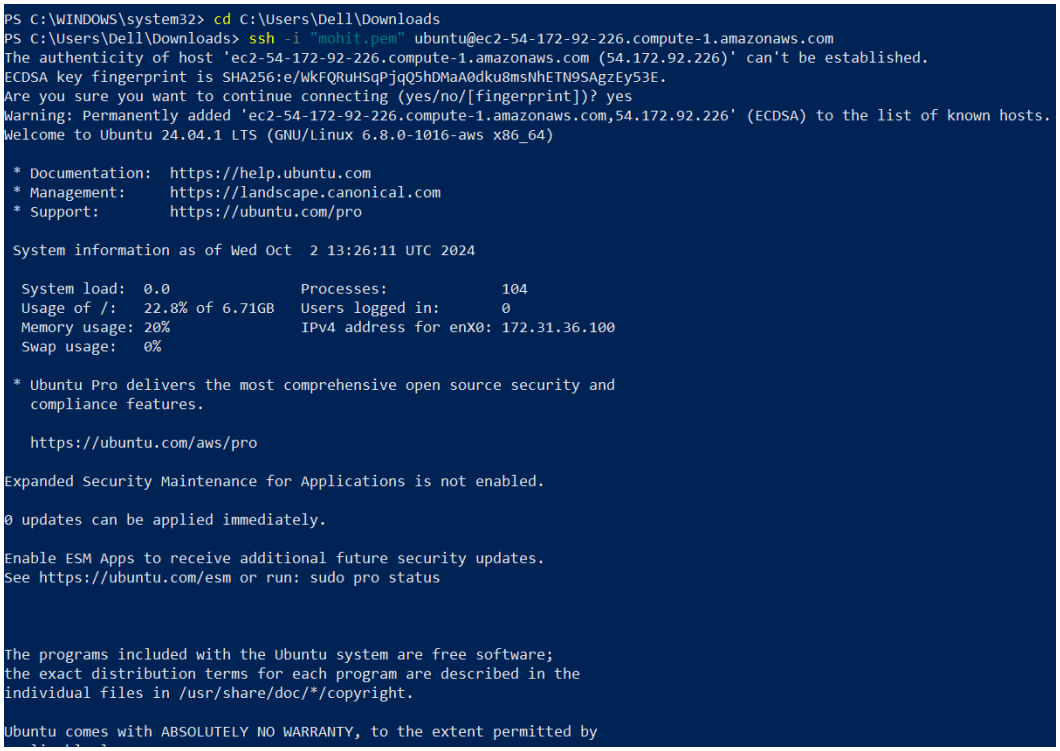
And by running sudo systemctl status nagios, we can again check whether our server is running or not



Step 4: TO BE DONE IN THE Nagios-client TERMINAL

Now it is time to switch to the client machine.

SSH into the machine or simply use the EC2 Instance Connect feature.



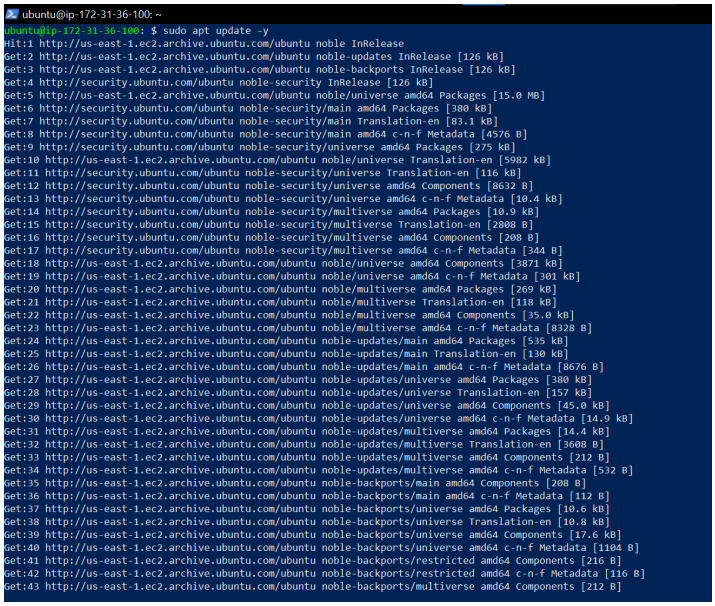
Make a package index update and install gcc, nagios-nrpe-server and the plugins. Run the following commands to achieve the same.

sudo apt update -y

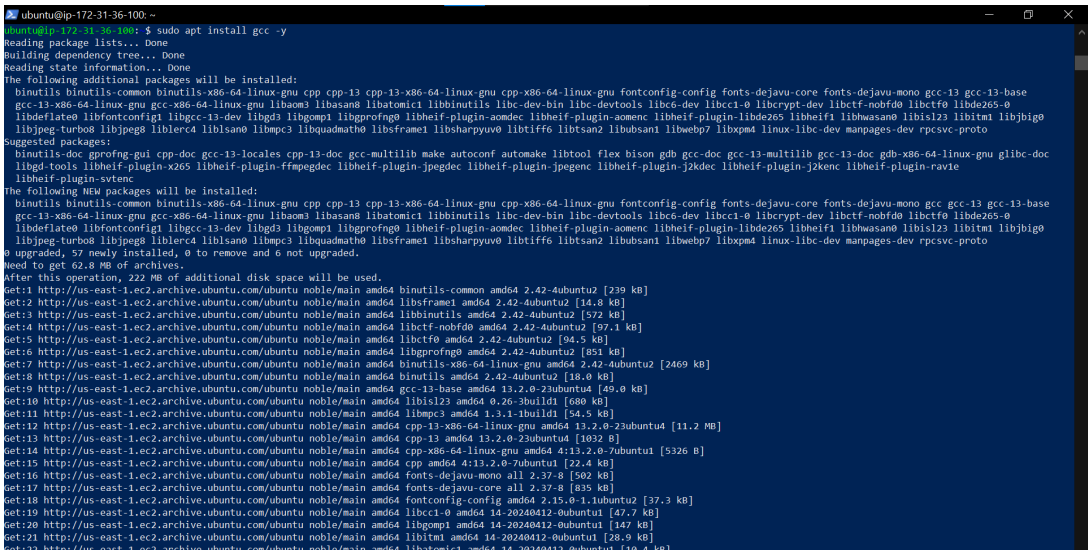
sudo apt install gcc -y

sudo apt install -y nagios-nrpe-server nagios-plugins

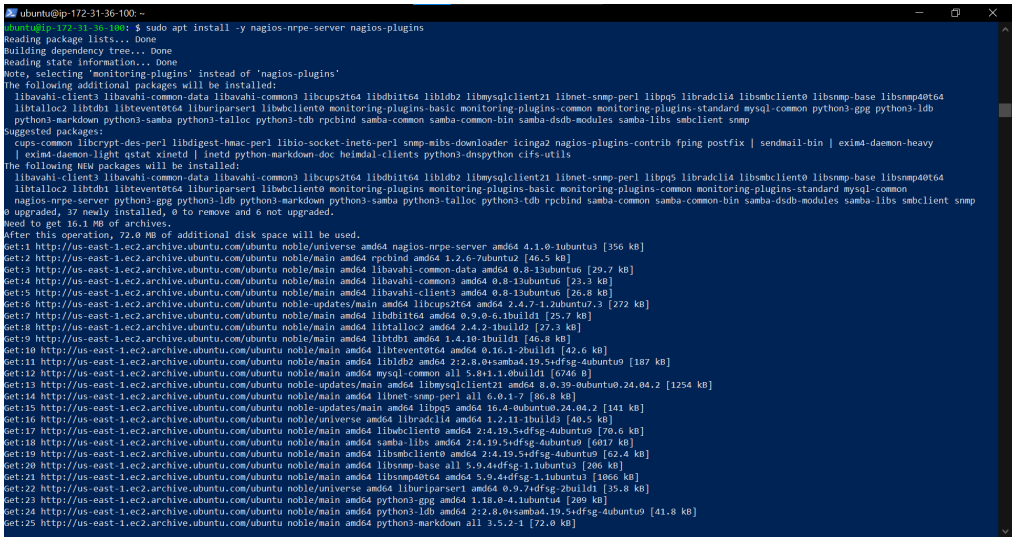
Name: Mohit Kerkar Div: D15C Roll No: 23



```
ubuntu@ip-172-31-36-100: ~
ubuntu@ip-172-31-36-100: $ sudo apt update -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [380 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [83.1 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4576 B]
Get:9 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [275 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [116 kB]
Get:12 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:13 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [10.4 kB]
Get:14 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [10.9 kB]
Get:15 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [2808 B]
Get:16 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:17 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [344 B]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [535 kB]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [130 kB]
Get:26 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [8676 B]
Get:27 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [380 kB]
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [157 kB]
Get:29 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [45.0 kB]
Get:30 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [14.9 kB]
Get:31 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [14.4 kB]
Get:32 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [3608 B]
Get:33 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [212 B]
Get:34 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [532 B]
Get:35 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:36 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [112 B]
Get:37 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [10.6 kB]
Get:38 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [10.8 kB]
Get:39 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [17.6 kB]
Get:40 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1104 B]
Get:41 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:42 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 c-n-f Metadata [116 B]
Get:43 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
```



```
ubuntu@ip-172-31-36-100: ~
ubuntu@ip-172-31-36-100: $ sudo apt install gcc -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu cpp cpp-13 cpp-13-x86-64-linux-gnu cpp-x86-64-linux-gnu fontconfig-config fonts-dejavu-core fonts-dejavu-mono gcc-13 gcc-13-base
  gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu libasan8 libatomic1 libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libde265-0
  libdeflate0 libfontconfig1 libgcc-13-dev libgd3 libgomp1 libgprofng0 libheif-plugin-aomdec libheif-plugin-aomenc libheif-plugin-libde265 libheif1 libhwasan0 libisl23 libitm1 libjbig0
  libjpeg-turbo8 libjpeg8 liblerc4 liblsan0 libmpc3 libquadmath0 libsframe1 libsharpyuv0 libtiff6 libtsan2 libubsan1 libwebp7 libxpm4 linux-libc-dev manpages-dev rpcsvc-proto
Suggested packages:
  binutils-doc gprofng-gui cpp-doc gcc-13-locales cpp-13-doc gcc-multilib make autoconf automake libtool flex bison gdb gcc-doc gcc-13-multilib gcc-13-doc gdb-x86-64-linux-gnu glibc-doc
  libgd-tools libheif-plugin-x265 libheif-plugin-ffmpegdec libheif-plugin-jpegdec libheif-plugin-jpegenc libheif-plugin-j2kdec libheif-plugin-j2kenc libheif-plugin-rav1e
  libheif-plugin-svtenc
The following NEW packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu cpp cpp-13 cpp-13-x86-64-linux-gnu cpp-x86-64-linux-gnu fontconfig-config fonts-dejavu-core fonts-dejavu-mono gcc gcc-13 gcc-13-base
  gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu libasan8 libatomic1 libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libde265-0
  libdeflate0 libfontconfig1 libgcc-13-dev libgd3 libgomp1 libgprofng0 libheif-plugin-aomdec libheif-plugin-aomenc libheif-plugin-libde265 libheif1 libhwasan0 libisl23 libitm1 libjbig0
  libjpeg-turbo8 libjpeg8 liblerc4 liblsan0 libmpc3 libquadmath0 libsframe1 libsharpyuv0 libtiff6 libtsan2 libubsan1 libwebp7 libxpm4 linux-libc-dev manpages-dev rpcsvc-proto
0 upgraded, 57 newly installed, 0 to remove and 6 not upgraded.
Need to get 62.8 MB of archives.
After this operation, 222 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 binutils-common amd64 2.42-4ubuntu2 [239 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libsframe1 amd64 2.42-4ubuntu2 [14.8 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libbinutils amd64 2.42-4ubuntu2 [572 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libctf-nobfd0 amd64 2.42-4ubuntu2 [97.1 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libctf0 amd64 2.42-4ubuntu2 [94.5 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libgprofng0 amd64 2.42-4ubuntu2 [851 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 binutils-x86-64-linux-gnu amd64 2.42-4ubuntu2 [2469 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 binutils amd64 2.42-4ubuntu2 [18.0 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 gcc-13-base amd64 13.2.0-23ubuntu4 [49.0 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libisl23 amd64 0.26-3build1 [680 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libmpc3 amd64 1.3.1-1build1 [54.5 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 cpp-13-x86-64-linux-gnu amd64 13.2.0-23ubuntu4 [11.2 MB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 cpp-13 amd64 13.2.0-23ubuntu4 [1032 B]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 cpp-x86-64-linux-gnu amd64 4:13.2.0-7ubuntu1 [5326 B]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 cpp amd64 4:13.2.0-7ubuntu1 [22.4 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 fonts-dejavu-mono all 2.37-8 [502 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 fonts-dejavu-core all 2.37-8 [835 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 fontconfig-config amd64 2.15.0-1.1ubuntu2 [37.3 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libcc1-0 amd64 14-20240412-0ubuntu1 [47.7 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libgomp1 amd64 14-20240412-0ubuntu1 [147 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libitm1 amd64 14-20240412-0ubuntu1 [28.9 kB]
```



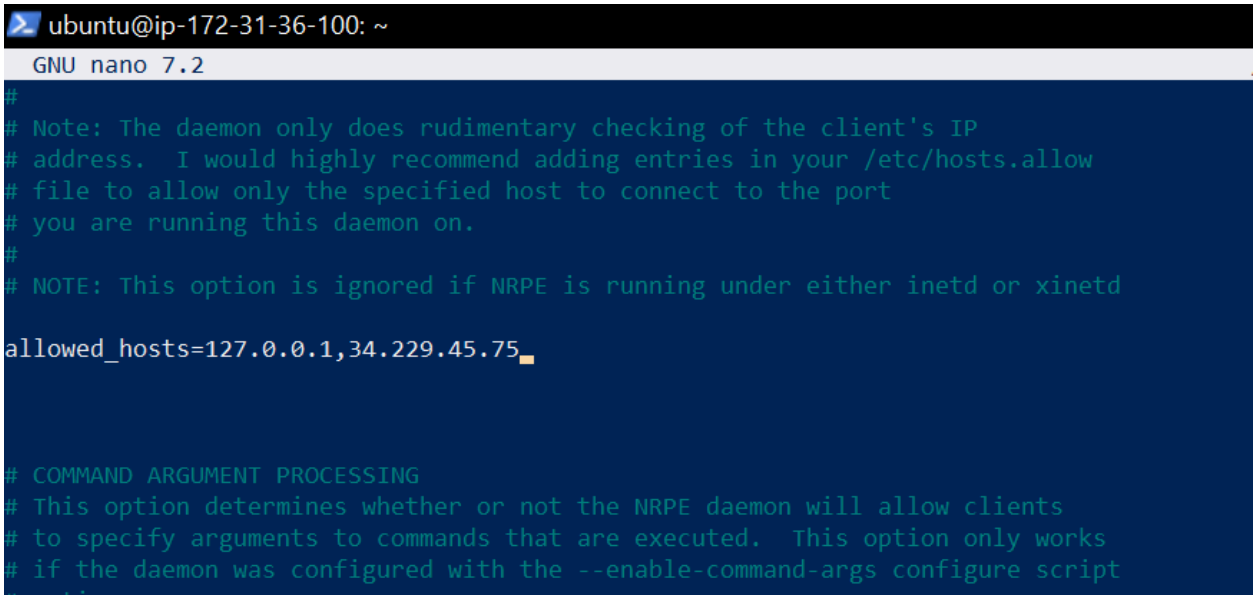
```
ubuntu@ip-172-31-36-100: ~
ubuntu@ip-172-31-36-100: $ sudo apt install -y nagios-nrpe-server nagios-plugins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'monitoring-plugins' instead of 'nagios-plugins'
The following additional packages will be installed:
  libavahi-client3 libavahi-common-data libavahi-common3 libcups2t64 libdbi1t64 libldb2 libmysqlclient21 libnet-snmp-perl libpq5 libradcli4 libsmbclient0 libsnmp-base libsnmp40t64
  libtalloc2 libtdb1 libtevent0t64 liburiparser1 libwbclient0 monitoring-plugins-basic monitoring-plugins-common monitoring-plugins-standard mysql-common python3-gpg python3-ldb
  python3-markdown python3-samba python3-talloc python3-tdb rpcbind samba-common samba-common-bin samba-dsdb-modules samba-libs smbclient snmp
Suggested packages:
  cups-common libcrypt-des-perl libdigest-hmac-perl libio-socket-inet6-perl snmp-mibs-downloader icinga2 nagios-plugins-contrib fping postfix | sendmail-bin | exim4-daemon-heavy
  | exim4-daemon-light qstat xinetd | inetd python-markdown-doc heimdal-clients python3-dnspython cifs-utils
The following NEW packages will be installed:
  libavahi-client3 libavahi-common-data libavahi-common3 libcups2t64 libdbi1t64 libldb2 libmysqlclient21 libnet-snmp-perl libpq5 libradcli4 libsmbclient0 libsnmp-base libsnmp40t64
  libtalloc2 libtdb1 libtevent0t64 liburiparser1 libwbclient0 monitoring-plugins monitoring-plugins-basic monitoring-plugins-common monitoring-plugins-standard mysql-common
  nagios-nrpe-server python3-gpg python3-ldb python3-markdown python3-samba python3-talloc python3-tdb rpcbind samba-common samba-common-bin samba-dsdb-modules samba-libs smbclient snmp
0 upgraded, 37 newly installed, 0 to remove and 6 not upgraded.
Need to get 16.1 MB of archives.
After this operation, 72.0 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 nagios-nrpe-server amd64 4.1.0-1ubuntu3 [356 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 rpcbind amd64 1.2.6-7ubuntu2 [46.5 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libavahi-common-data amd64 0.8-13ubuntu6 [29.7 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libavahi-common3 amd64 0.8-13ubuntu6 [23.3 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libavahi-client3 amd64 0.8-13ubuntu6 [26.8 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libcups2t64 amd64 2.4.7-1.2ubuntu7.3 [272 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libdbi1t64 amd64 0.9.0-6.1build1 [25.7 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libtalloc2 amd64 2.4.2-1build2 [27.3 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libtdb1 amd64 1.4.10-1build1 [46.8 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libtevent0t64 amd64 0.16.1-2build1 [42.6 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libldb2 amd64 2:2.8.0+samba4.19.5+dfsg-4ubuntu9 [187 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 mysql-common all 5.8+1.1.0build1 [6746 B]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libmysqlclient21 amd64 8.0.39-0ubuntu0.24.04.2 [1254 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libnet-snmp-perl all 6.0.1-7 [86.8 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libpq5 amd64 16.4-0ubuntu0.24.04.2 [141 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 libradcli4 amd64 1.2.11-1build3 [40.5 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libwbclient0 amd64 2:4.19.5+dfsg-4ubuntu9 [70.6 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 samba-libs amd64 2:4.19.5+dfsg-4ubuntu9 [6017 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libsmbclient0 amd64 2:4.19.5+dfsg-4ubuntu9 [62.4 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libsnmp-base all 5.9.4+dfsg-1.1ubuntu3 [206 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libsnmp40t64 amd64 5.9.4+dfsg-1.1ubuntu3 [1066 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 liburiparser1 amd64 0.9.7+dfsg-2build1 [35.8 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 python3-gpg amd64 1.18.0-4.1ubuntu4 [209 kB]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 python3-ldb amd64 2:2.8.0+samba4.19.5+dfsg-4ubuntu9 [41.8 kB]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 python3-markdown all 3.5.2-1 [72.0 kB]
```

Name: Mohit Kerkar Div: D15C Roll No: 23

Open nrpe.cfg file to make changes.

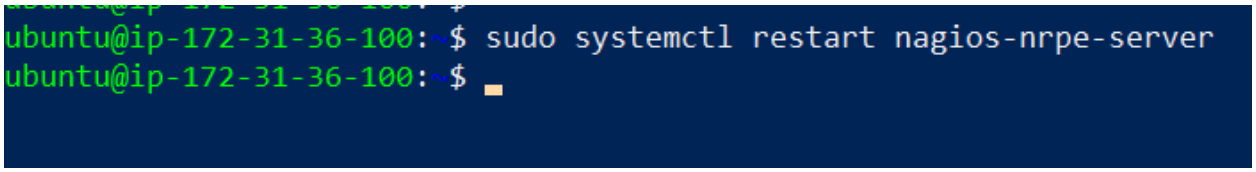
sudo nano /etc/nagios/nrpe.cfg

Under allowed_hosts, add your nagios host IP address like so



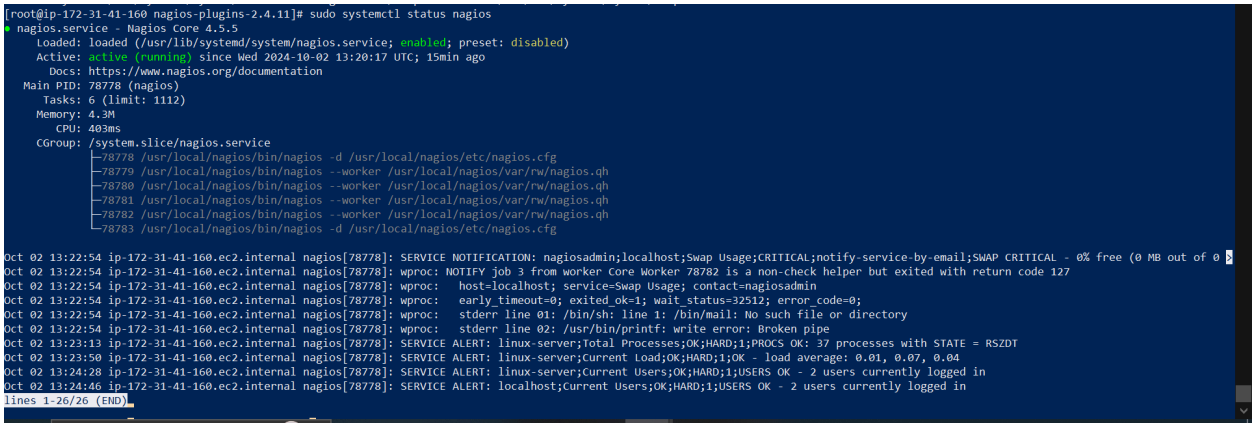
Now restart the NRPE server by this command.

sudo systemctl restart nagios-nrpe-server



Run the following command in the Nagios-host terminal

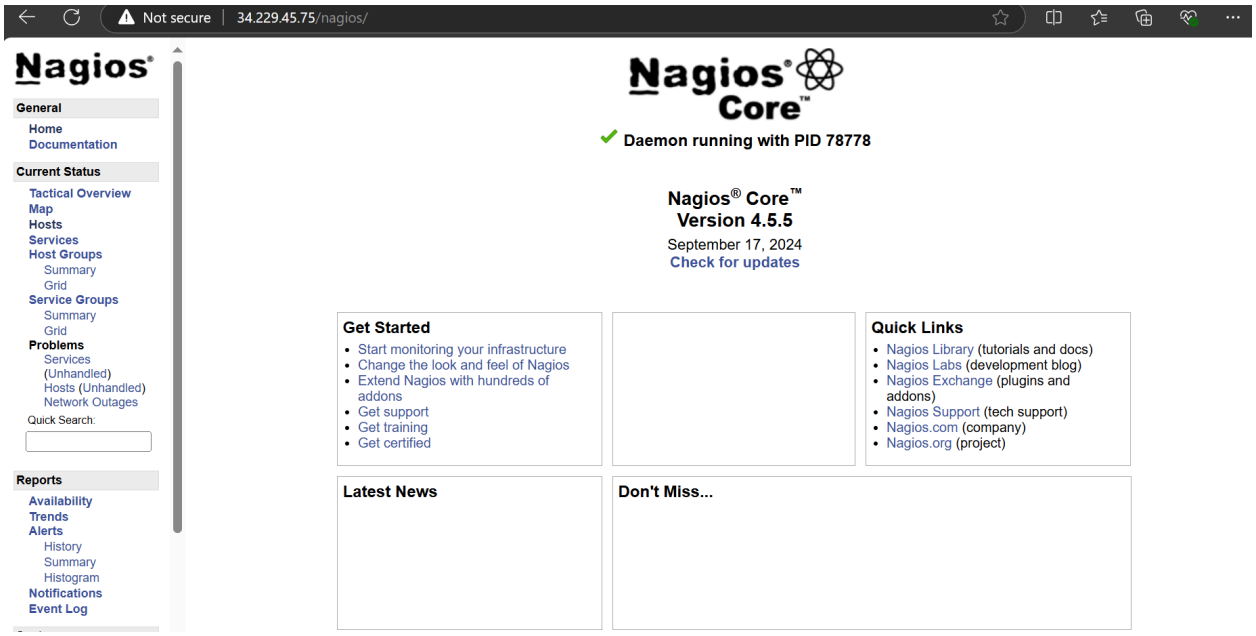
sudo systemctl status nagios



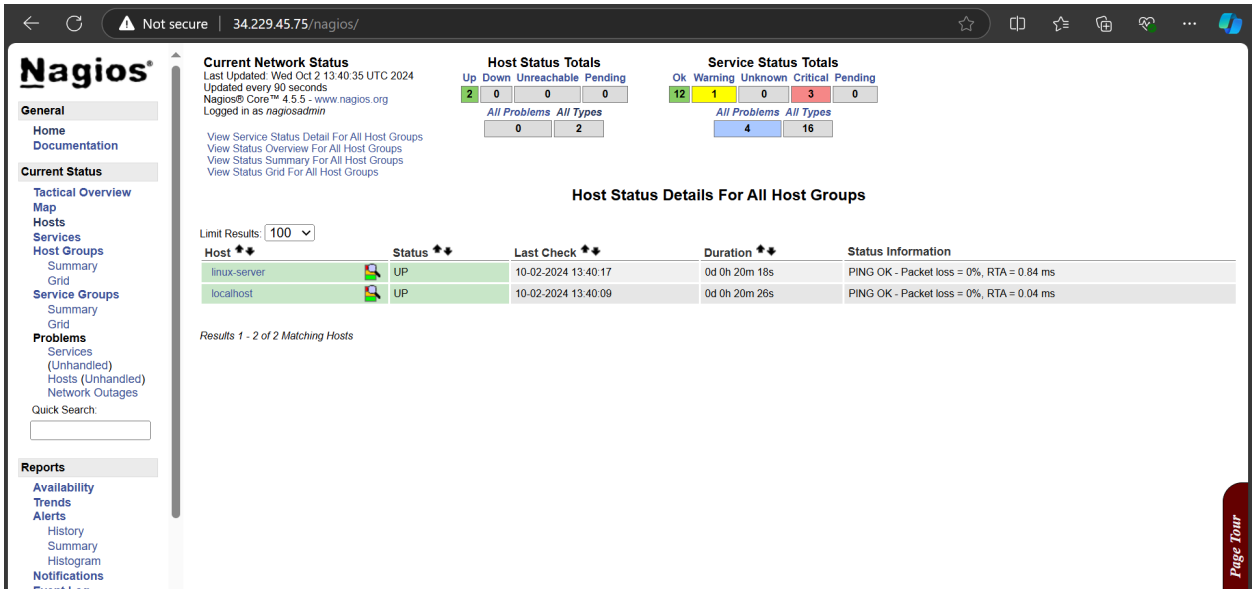
Name: Mohit Kerkar Div: D15C Roll No: 23

Step 5: Visiting your nagios server using your nagios-host ip address

Open up your browser and look for http://<public_ip_address_of_nagios-host>/nagios

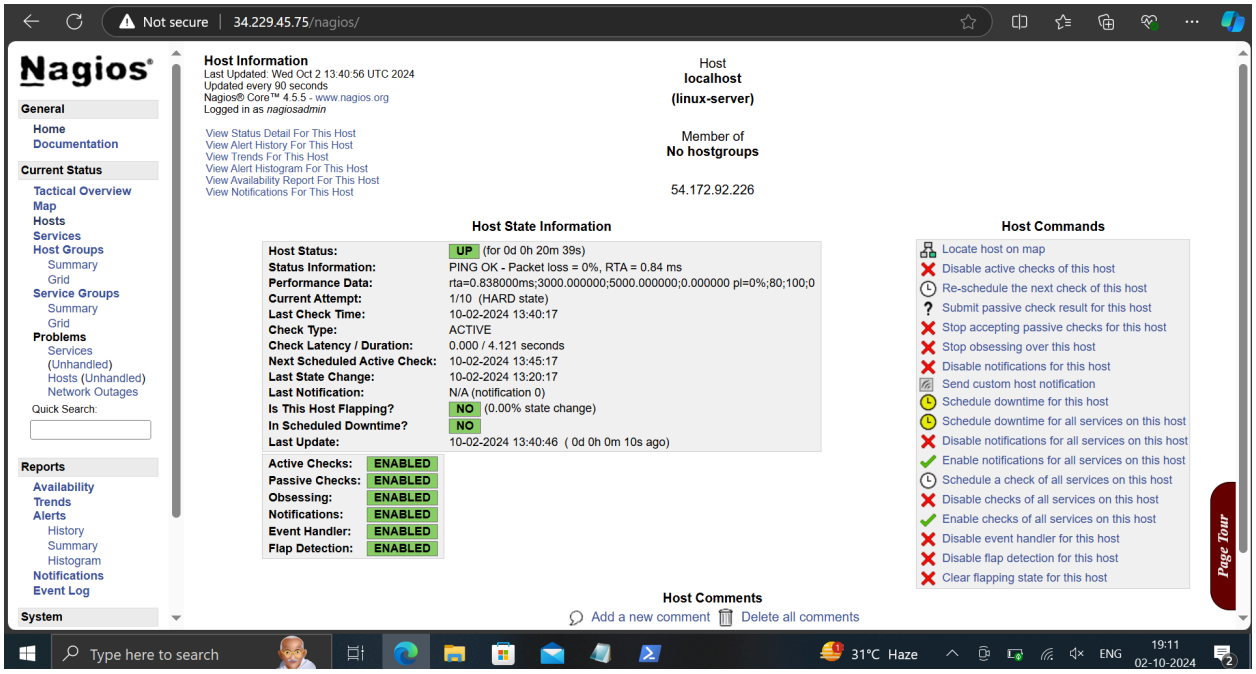


Click on Hosts.

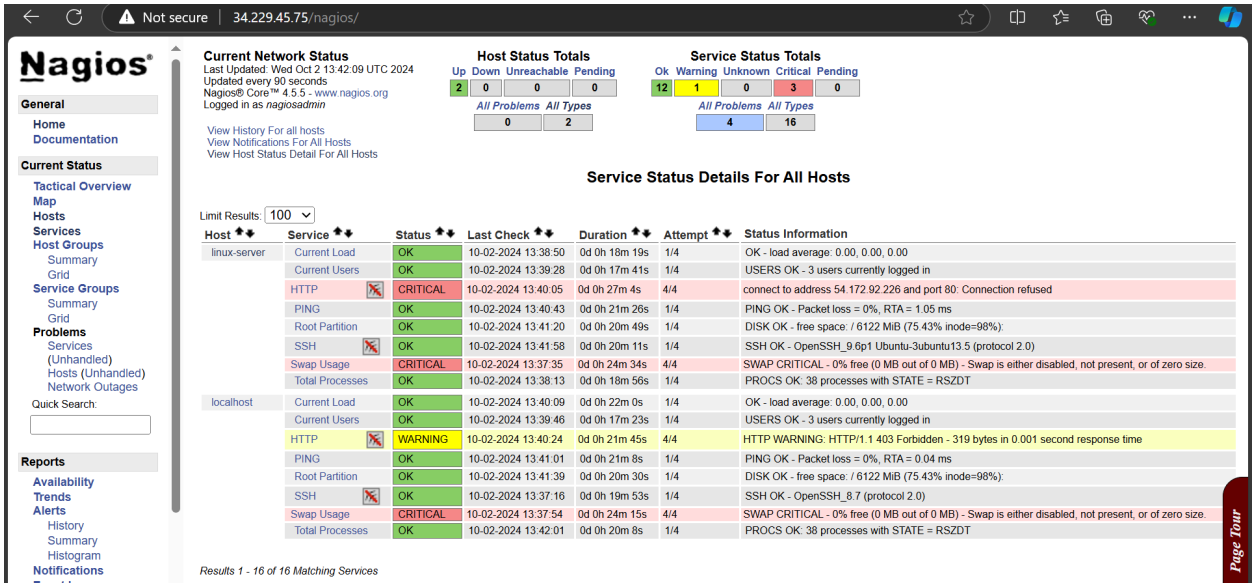


Name: Mohit Kerkar Div: D15C Roll No: 23

Click on linux-server to view host information



We can even navigate to the services section, which explicitly mentions the status, duration, checks, information about the numerous services present on our hosts



Conclusion: In conclusion, the experiment focused on monitoring ports, services, and a Linux server using Nagios. Through the step-by-step process, we successfully configured Nagios to monitor essential network services on the Linux server. By setting up both the Nagios host and client, we were able to track system performance, ensure service availability, and monitor key metrics like CPU and memory usage

Name: Mohit Kerkar Div: D15C Roll No: 23

Adv DevOps Exp 11

Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

Theory:

AWS Lambda

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). Users of AWS Lambda create functions, self-contained applications written in one of the supported languages and runtimes, and upload them to AWS Lambda, which executes those functions in an efficient and flexible manner.The Lambda functions can perform any kind of computing task, from serving web pages and processing streams of data to calling APIs and integrating with other AWS services. The concept of “serverless” computing refers to not needing to maintain your own servers to run these functions. AWS Lambda is a fully managed service that takes care of all the infrastructure for you. And so “serverless” doesn’t mean that there are no servers involved: it just means that the servers, the operating systems, the network layer and the rest of the infrastructure have already been taken care of so that you can focus on writing application code.

Features of AWS Lambda

- AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved.
- It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateway, Kinesis, CodeCommit, and many more to trigger an event.
- You don’t need to invest upfront. You pay only for the memory used by the lambda function and minimal cost on the number of requests hence cost-efficient.
- AWS Lambda is secure. It uses AWS IAM to define all the roles and security policies.
- It offers fault tolerance for both services running the code and the function. You do not have to worry about the application down.

Packaging Functions

Lambda functions need to be packaged and sent to AWS. This is usually a process of compressing the function and all its dependencies and uploading it to an S3 bucket. And letting AWS know that you want to use this package when a specific event takes place. To help us with this process we use the Serverless Stack Framework (SST). We’ll go over this in detail later on in
this guide.

Execution Model

The container (and the resources used by it) that runs our function is managed completely by AWS. It is brought up when an event takes place and is turned off if it is not being used. If additional requests are made while the original event is being served, a new container is brought
up to serve a request. This means that if we are undergoing a usage spike, the cloud provider simply creates multiple instances of the container with our function to serve those requests. This has some interesting implications. Firstly, our functions are effectively stateless. Secondly,

each request (or event) is served by a single instance of a Lambda function. This means that you
are not going to be handling concurrent requests in your code. AWS brings up a container whenever there is a new request. It does make some optimizations here. It will hang on to the container for a few minutes (5 - 15mins depending on the load) so it can respond to subsequent requests without a cold start.

Stateless Functions

The above execution model makes Lambda functions effectively stateless. This means that every
time your Lambda function is triggered by an event it is invoked in a completely new environment. You don't have access to the execution context of the previous event.
However, due to the optimization noted above, the actual Lambda function is invoked only once per container instantiation. Recall that our functions are run inside containers. So when a function is first invoked, all the code in our handler function gets executed and the handler function gets invoked. If the container is still available for subsequent requests, your function will get invoked and not the code around it.
For example, the createNewDbConnection method below is called once per container instantiation and not every time the Lambda function is invoked. The myHandler function on the other hand is called on every invocation.

Common Use Cases for Lambda

Due to Lambda's architecture, it can deliver great benefits over traditional cloud computing setups for applications where:
1. Individual tasks run for a short time;
2. Each task is generally self-contained;
3. There is a large difference between the lowest and highest levels in the workload of the application.
Some of the most common use cases for AWS Lambda that fit these criteria are: Scalable APIs. When building APIs using AWS Lambda, one execution of a Lambda function can serve a single HTTP request. Different parts of the API can be routed to different Lambda functions via Amazon API Gateway. AWS Lambda automatically scales individual functions according to

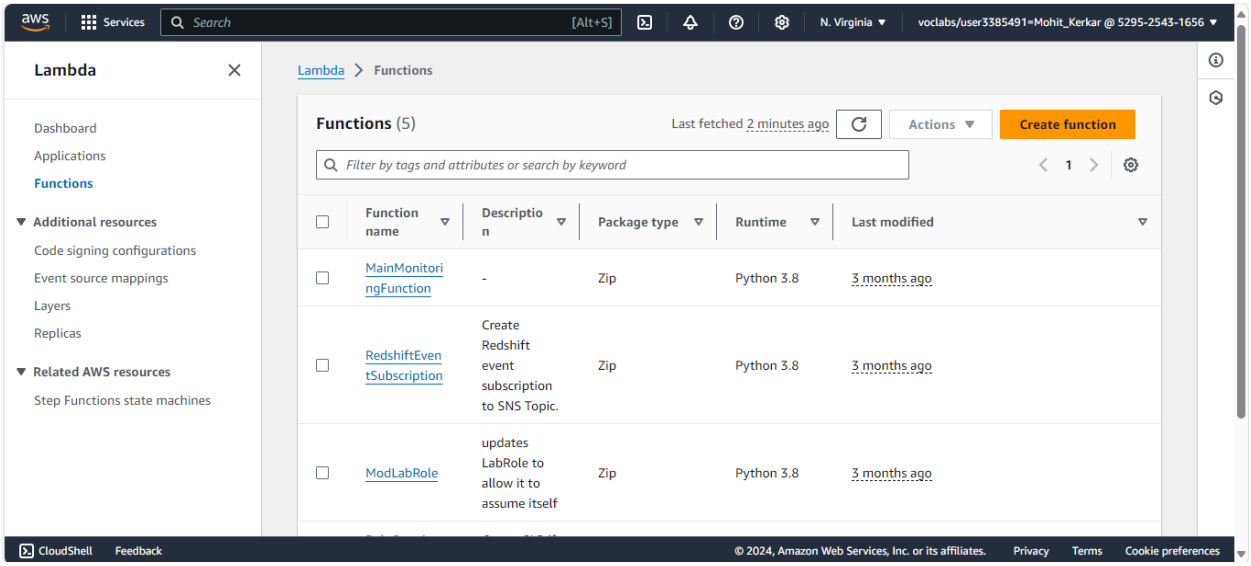
the demand for them, so different parts of your API can scale differently according to current usage levels. This allows for cost-effective and flexible API setups.
Data processing. Lambda functions are optimized for event-based data processing. It is easy to integrate AWS Lambda with data sources like Amazon DynamoDB and trigger a Lambda function for specific kinds of data events. For example, you could employ Lambda to do some work every time an item in DynamoDB is created or updated, thus making it a good fit for things like notifications, counters and analytics.

Name: Mohit Kerkar Div: D15C Roll No: 23

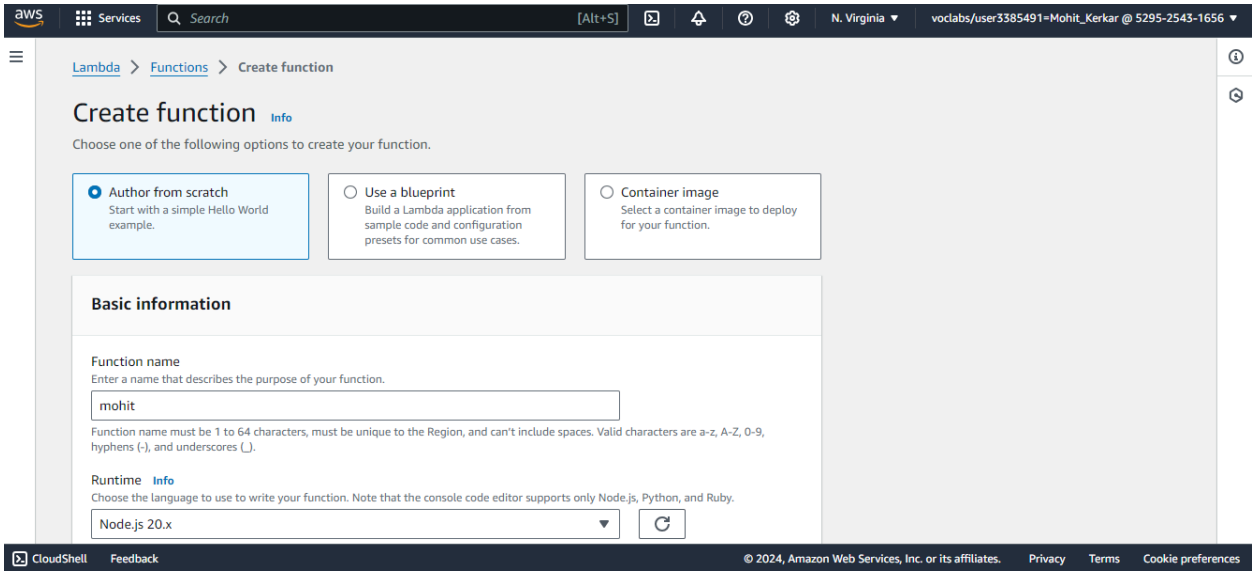
Steps to create and test your lambda function are as follows:-

Step 1: Creating a Function inside lambda

Navigate to Lambda inside AWS services and click on ‘Create Function’ button to create a new function inside Lambda services

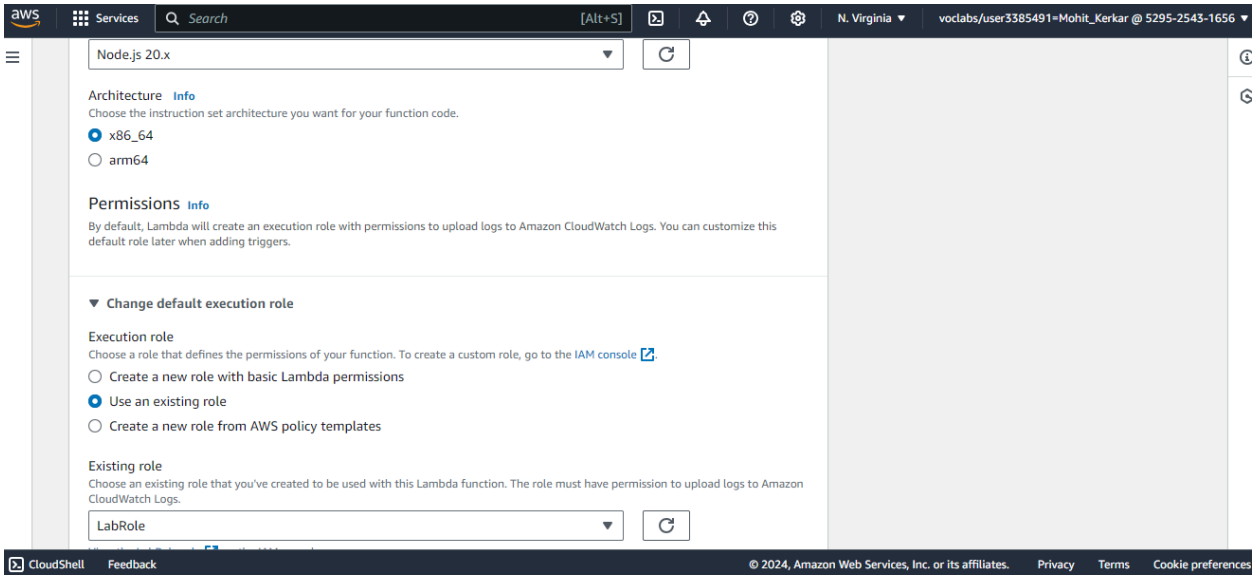


Inside the Create function process, we are supposed to enter the details such as the name of the function, details regarding the things that we to want the function to run or test (eg. programming language).

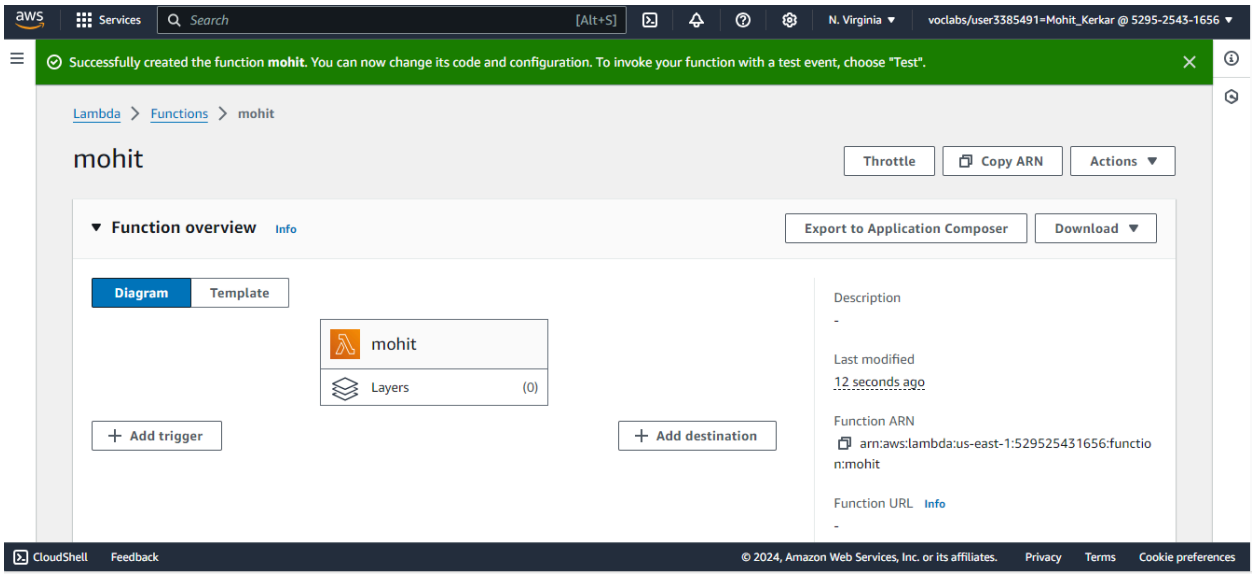


Along with these things, we are also required to set permissions for us to access and use the lambda function efficiently. For that purpose, we are to select ‘Use an existing role’ option and

select ‘LabRole’ which is an already existing role with permissions to use the Lambda Service.



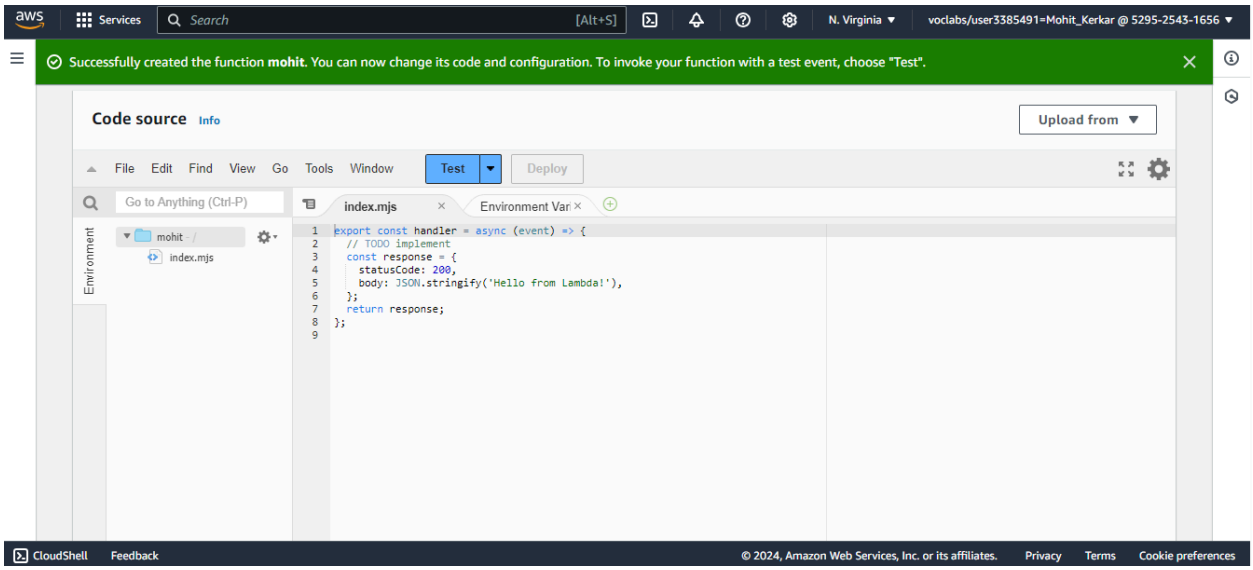
And that is how our function inside Lambda is successfully created. This is what appears on the screen after creation of our function (Our dashboard)



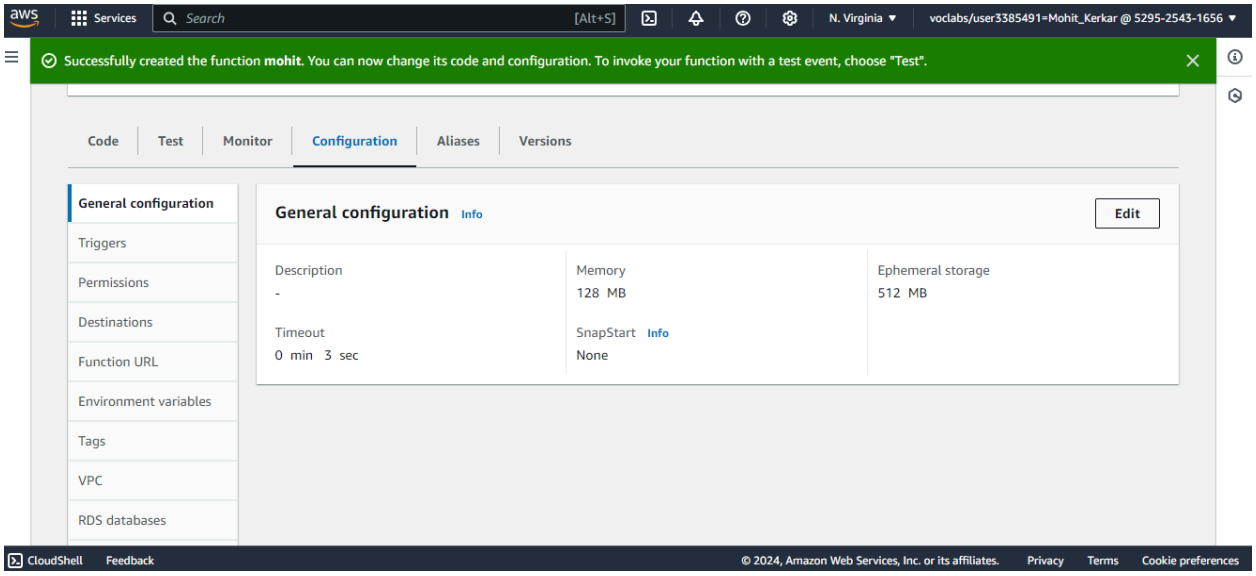
Step 2: Modifying function’s configuration

Name: Mohit Kerkar Div: D15C Roll No: 23

Scroll down to view the default code given in the index.mjs file as we see in the screenshot attached below

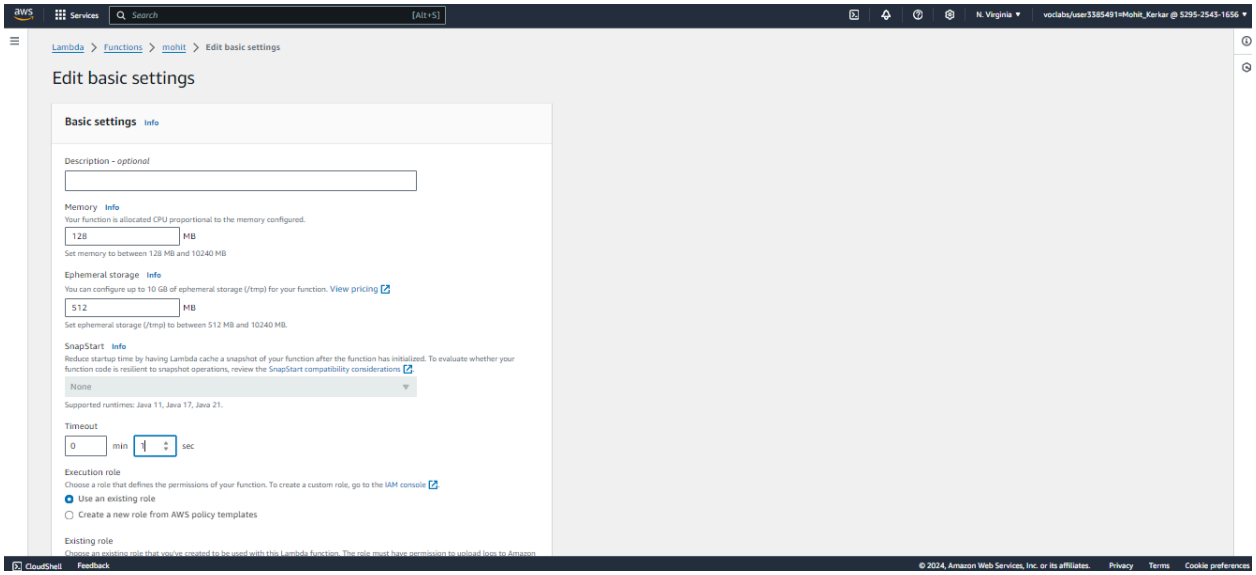


Navigate to the configuration section of our function. This section provides us details about the settings that have been applied to our function for carrying out functionalities like testing on our code in the language that we specified earlier

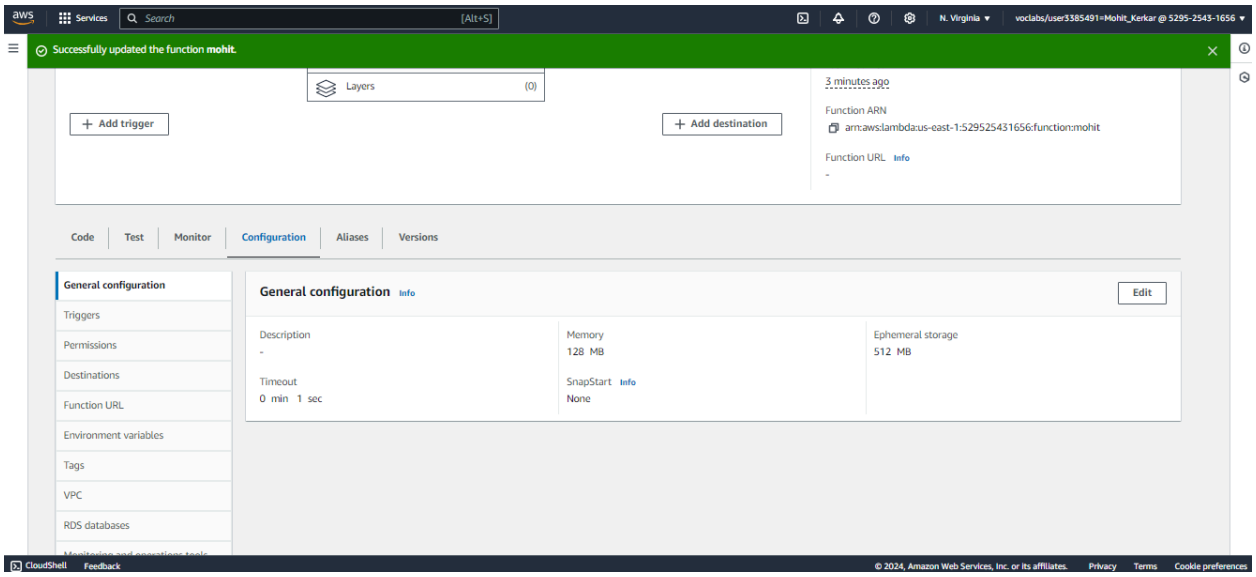


Name: Mohit Kerkar Div: D15C Roll No: 23

Click on Edit. Change the timeout to 1 sec to modify the time for the function to be kept running before forcibly terminating it.
After changing the configuration, click on save.



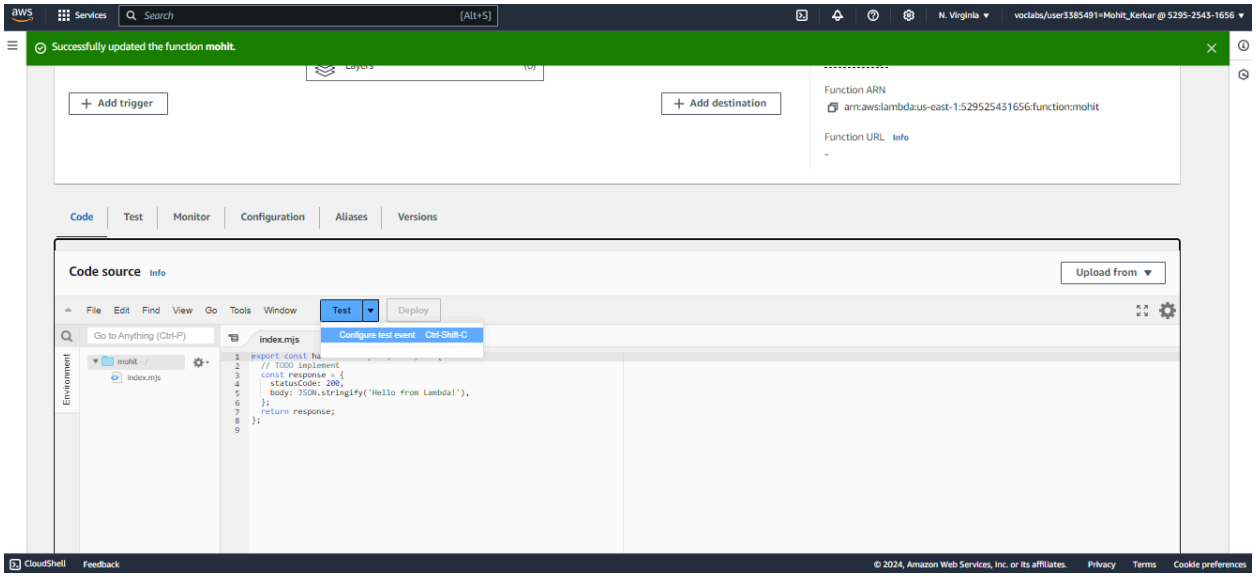
After saving, the configuration setting changes are reflected in the ‘General configuration’ section.



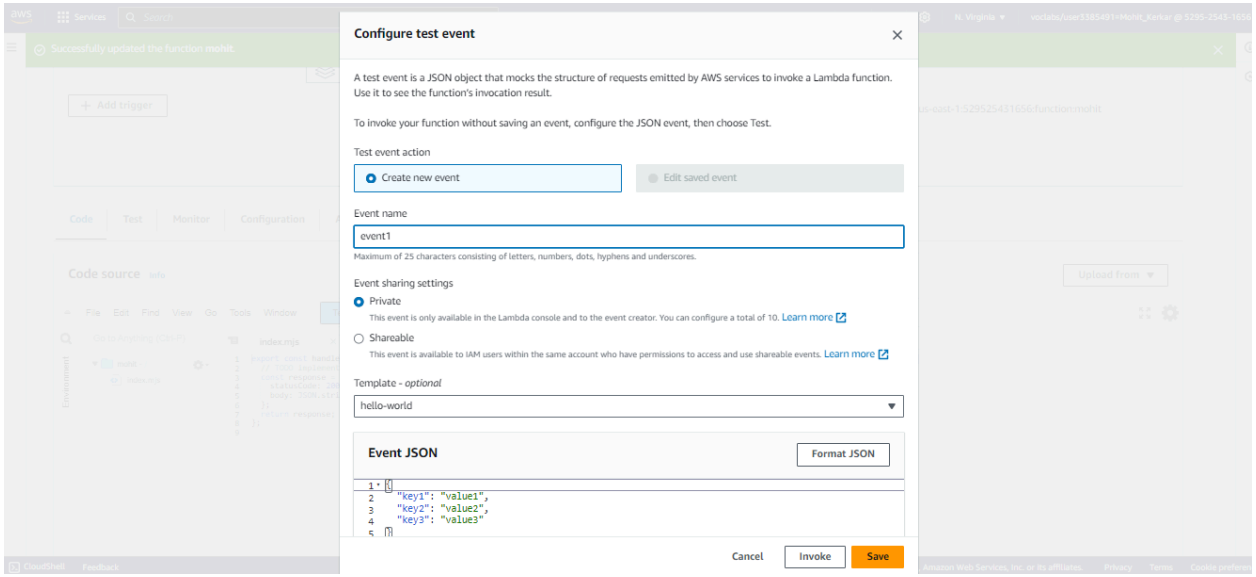
Name: Mohit Kerkar Div: D15C Roll No: 23

Step 3: Creating a new Test event

In our code section, we are given an option to test our code. For this purpose, we are offered numerous test events with which we can possibly test our code.



Click on ‘configure test event’ option for making a new event. Give a suitable name to the event and let other settings be as they are…

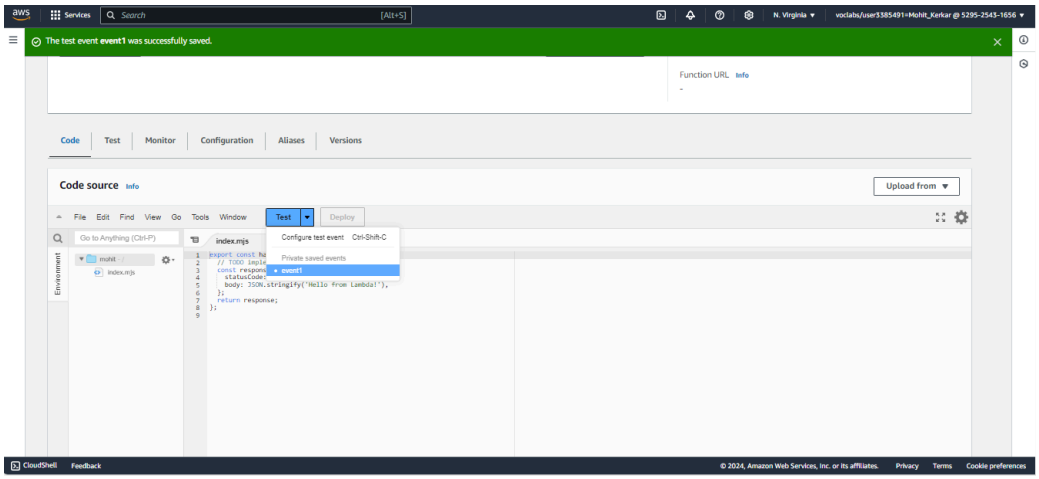


Click on Save..

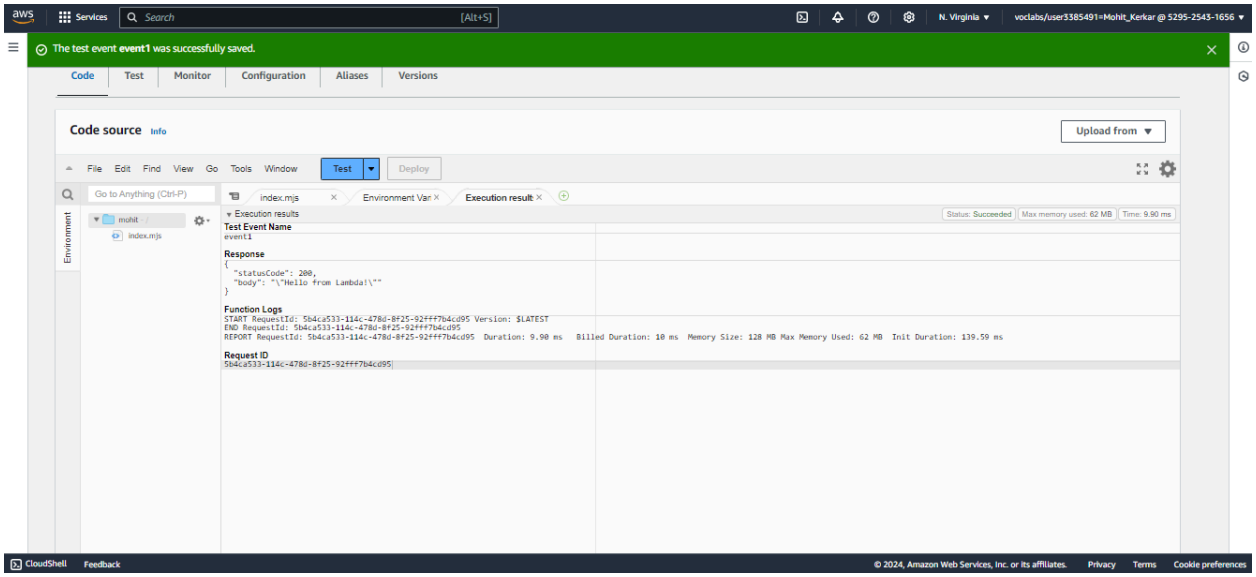
Name: Mohit Kerkar Div: D15C Roll No: 23

Step 4: Test with the new test event that we created

Now, in order to test the code according to the test event that we built in the previous step, again select the Test option and within we will be able to see that test event's name (for me it is event1). Select it and observe the changes that happen



There will be another tab opening which gives us our result after execution. Following is the format with which our Test result comes…

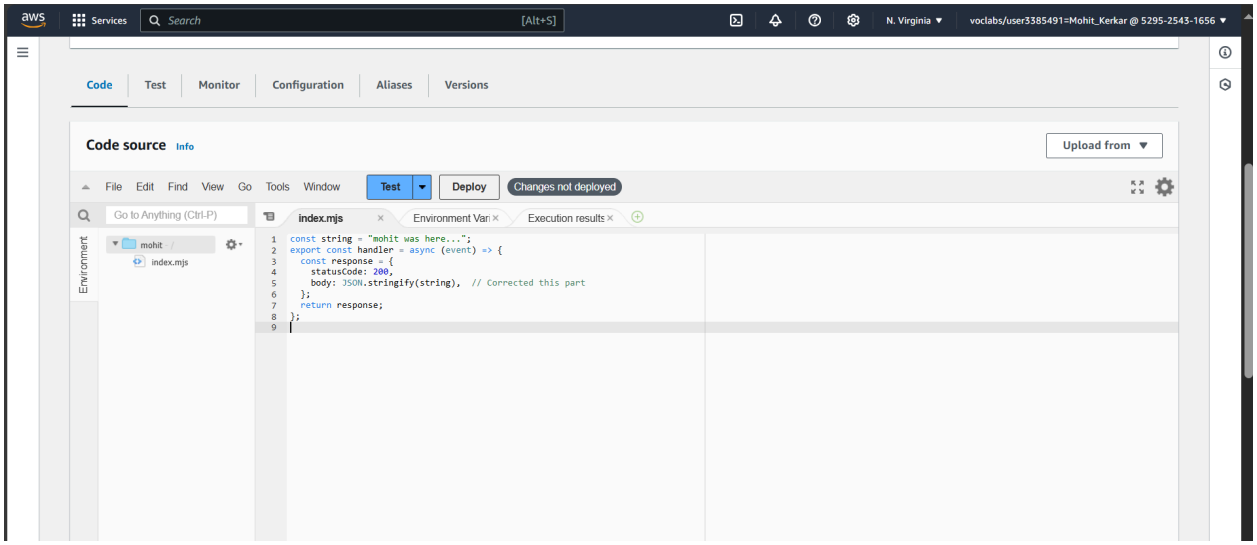


Name: Mohit Kerkar Div: D15C Roll No: 23

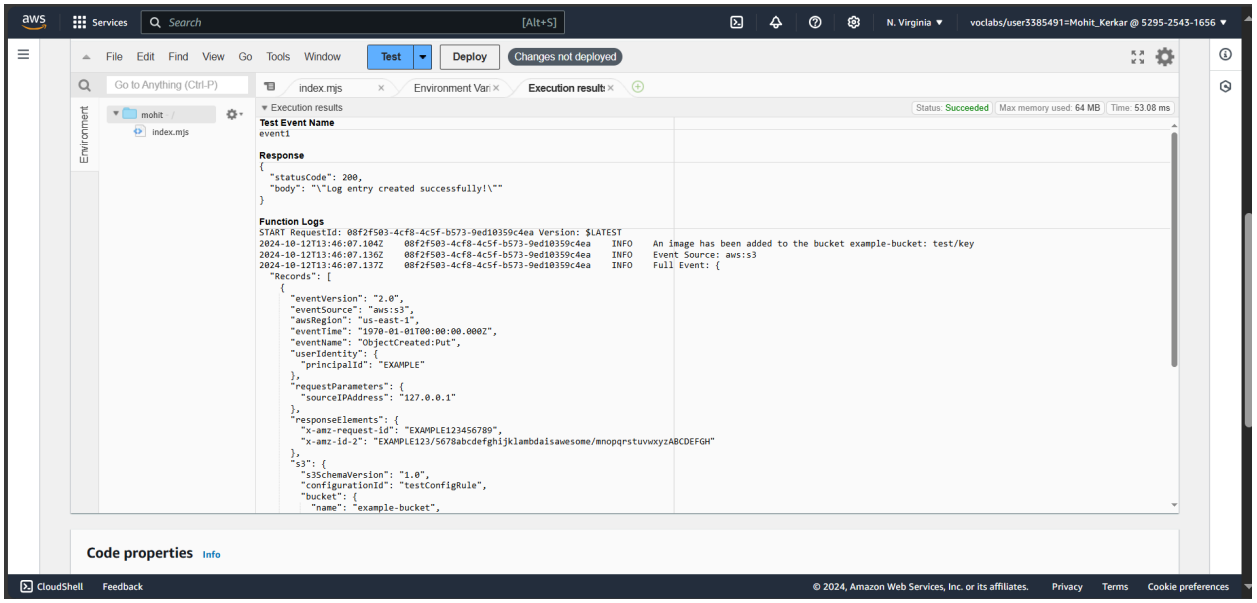
Step 5: Testing our modified code

Now, i added a new line at the start of my index.mjs file in my lambda function

Const string=”mohit was here…”



The execution result displays that string result in the output as well ..



Conclusion: In conclusion, AWS Lambda provides an efficient, serverless computing service that enables you to run code without managing infrastructure. By creating and configuring a Lambda function, you can execute code in response to various events, such as S3 uploads or API requests. The setup process involves defining roles, setting up test events, and modifying the function’s configuration for specific behaviors like timeouts. AWS Lambda’s stateless nature and automatic scaling make it ideal for short-lived, event-driven tasks, such as building APIs or processing data streams, ensuring cost-effective and scalable solutions.

Name: Mohit Kerkar Div: D15C Roll No: 23

Adv DevOps Exp 12

Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3

Theory:

AWS Lambda and S3 Integration:

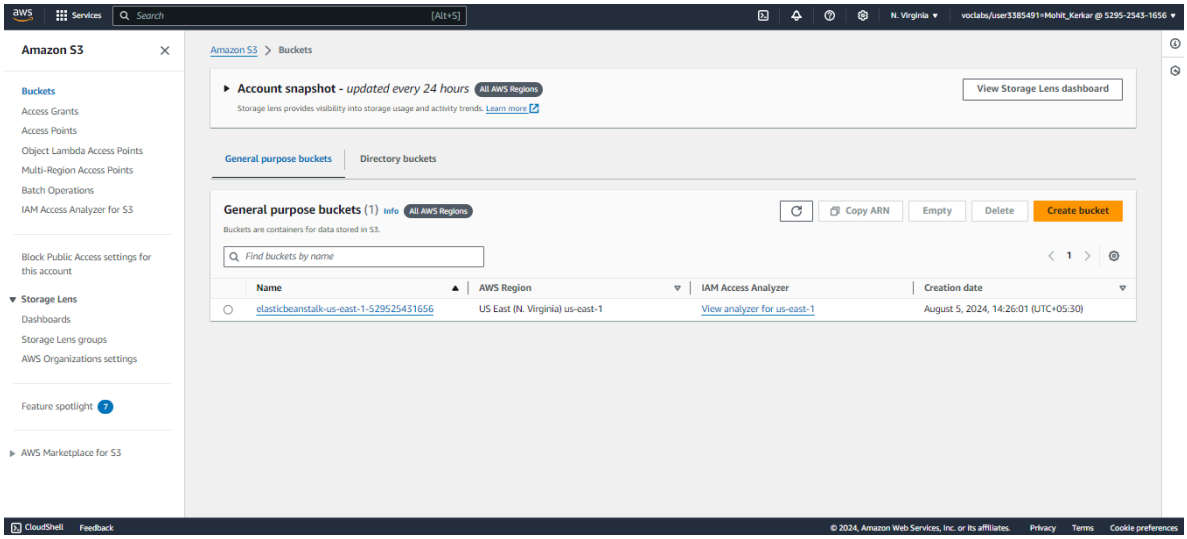
AWS Lambda allows you to execute code in response to various events, including those triggered by Amazon S3. When an object is added to an S3 bucket, it can trigger a Lambda function to execute, allowing for event-driven processing without managing servers.

Workflow:

1. Create an S3 Bucket:
○ First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.
2. Create the Lambda Function:
○ Set up a new Lambda function using AWS Lambda’s console. You can choose a runtime environment like Python, Node.js, or Java.
○ Write code that logs a message like “An Image has been added” when triggered. 3. Set Up Permissions:
○ Ensure that the Lambda function has the necessary permissions to access S3. You can do this by attaching an IAM role with policies that allow reading from the bucket and writing logs to CloudWatch.
4. Configure S3 Trigger:
○ Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).
5. Test the Setup:
○ Upload an object (e.g., an image) to the S3 bucket to test the trigger. The Lambda function should execute and log the message “An Image has been added” in AWS CloudWatch Logs.

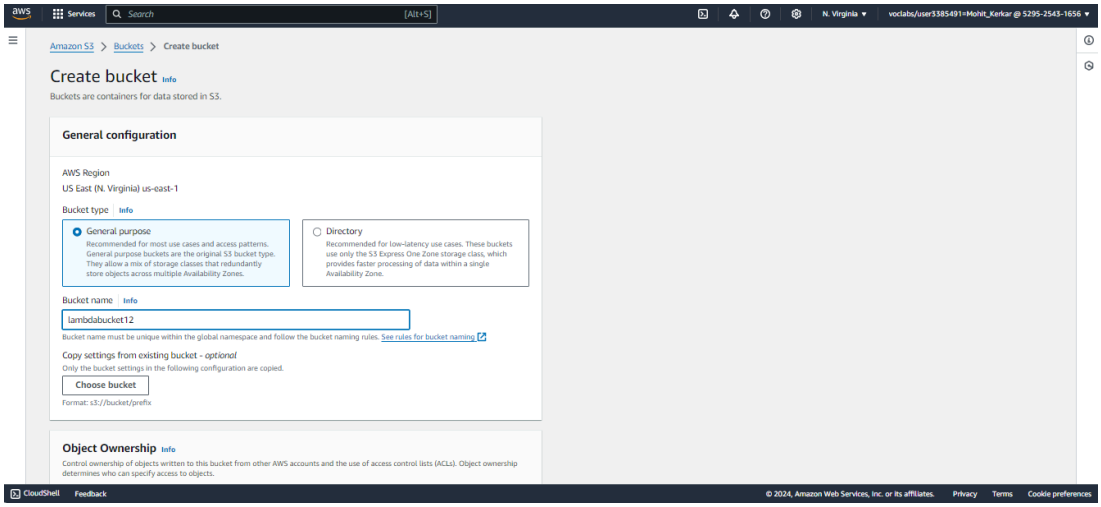
Step 1: S3 bucket creation

Navigate to the S3 inside services section on AWS and create a new bucket

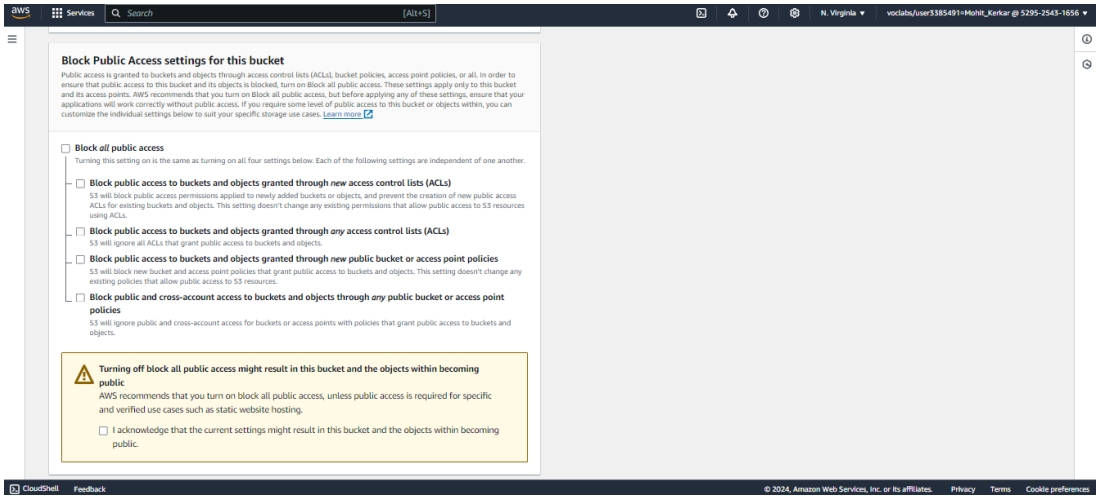


Name: Mohit Kerkar Div: D15C Roll No: 23

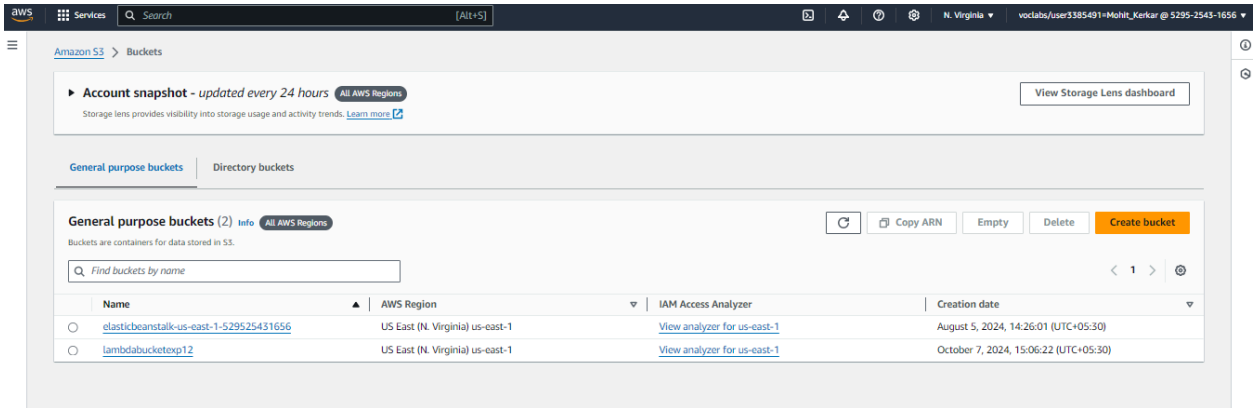
Give some suitable name to the bucket



Unselect all of the default settings related to ‘Block public access’



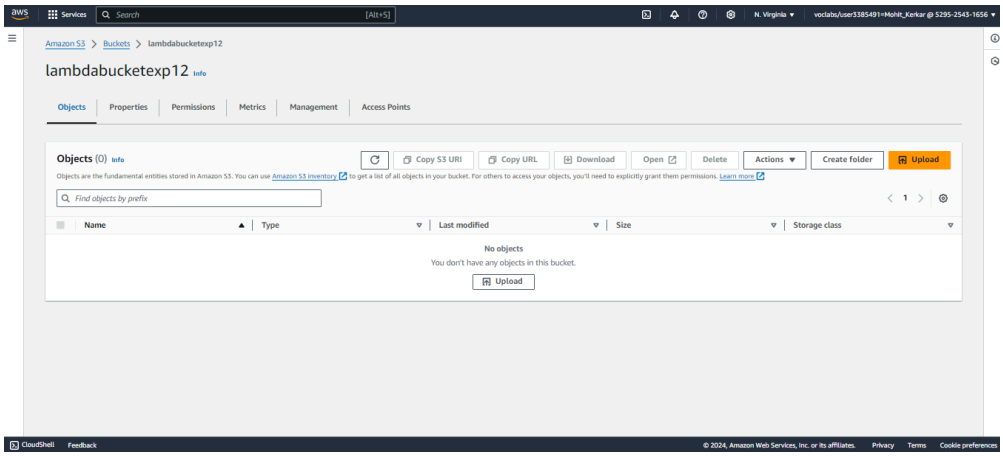
And here, you have your new S3 bucket created…



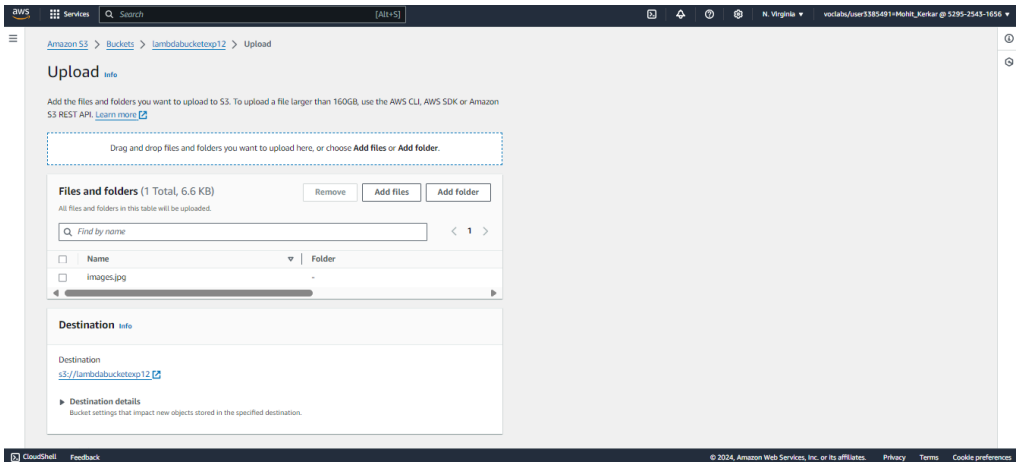
Name: Mohit Kerkar Div: D15C Roll No: 23

Step 2: Upload an image onto our bucket

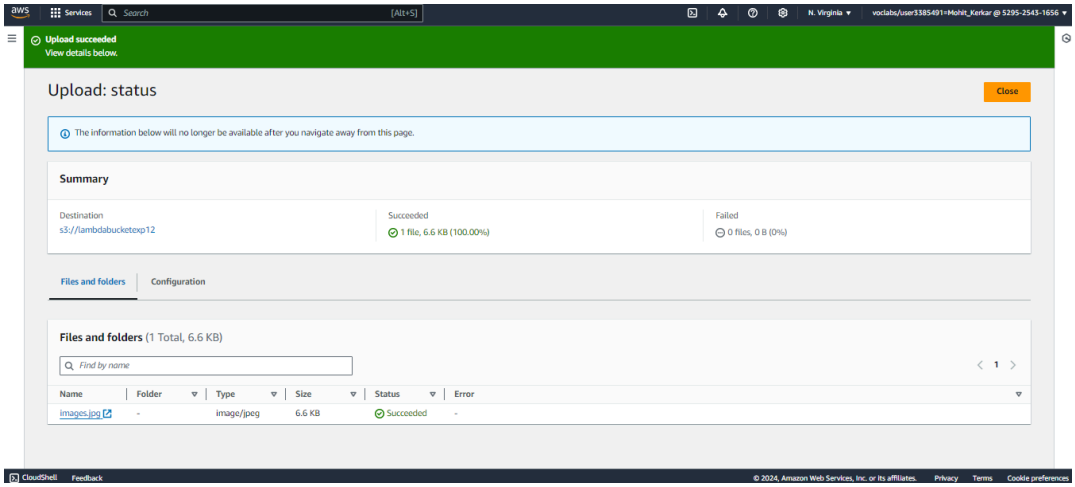
Now, open the bucket and click on upload



Upload a random saved image from your device onto the bucket by clicking on ‘Add files’ inside the bucket



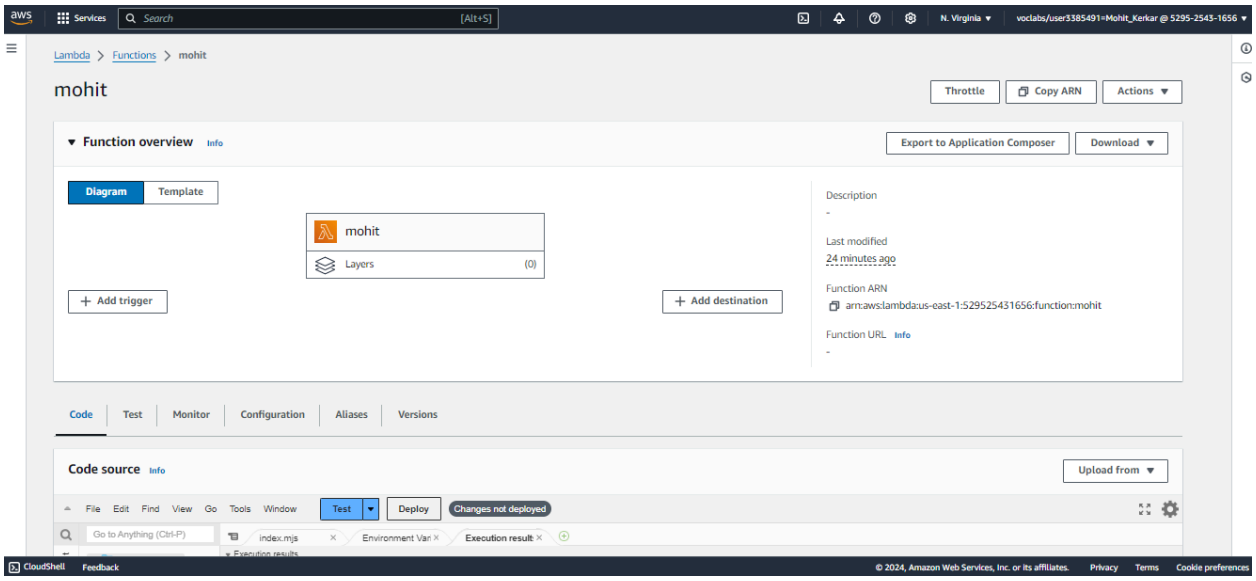
After uploading the image onto the bucket, this is the screen that will appear…



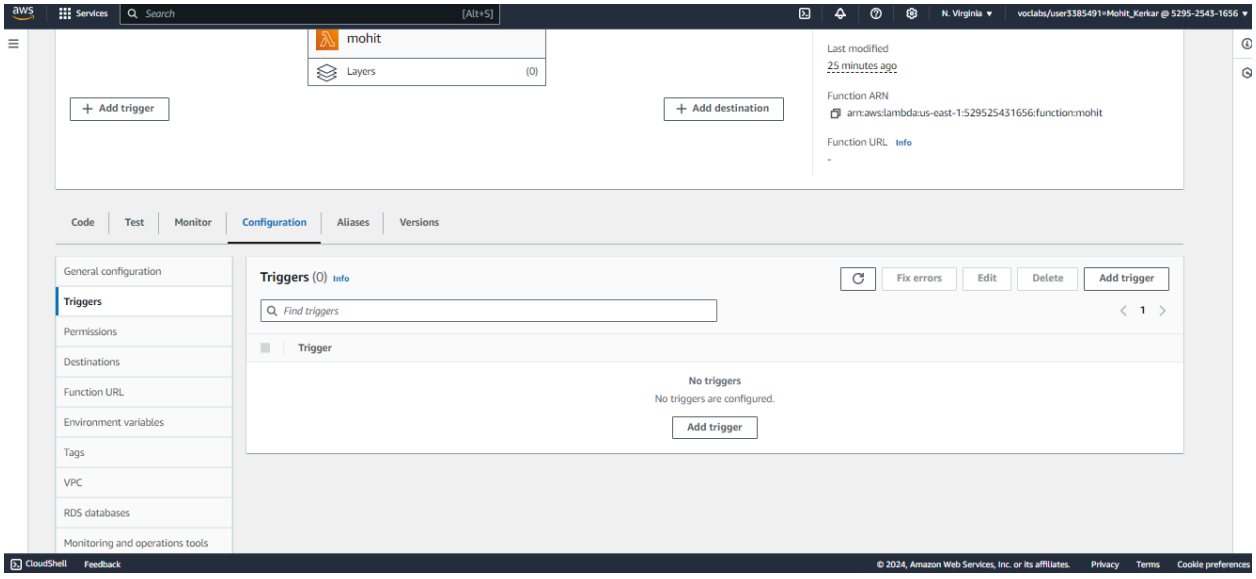
Name: Mohit Kerkar Div: D15C Roll No: 23

Step 3: Modifying Triggers inside Lambda configuration

Go back to the lambda function that we created in the previous experiment

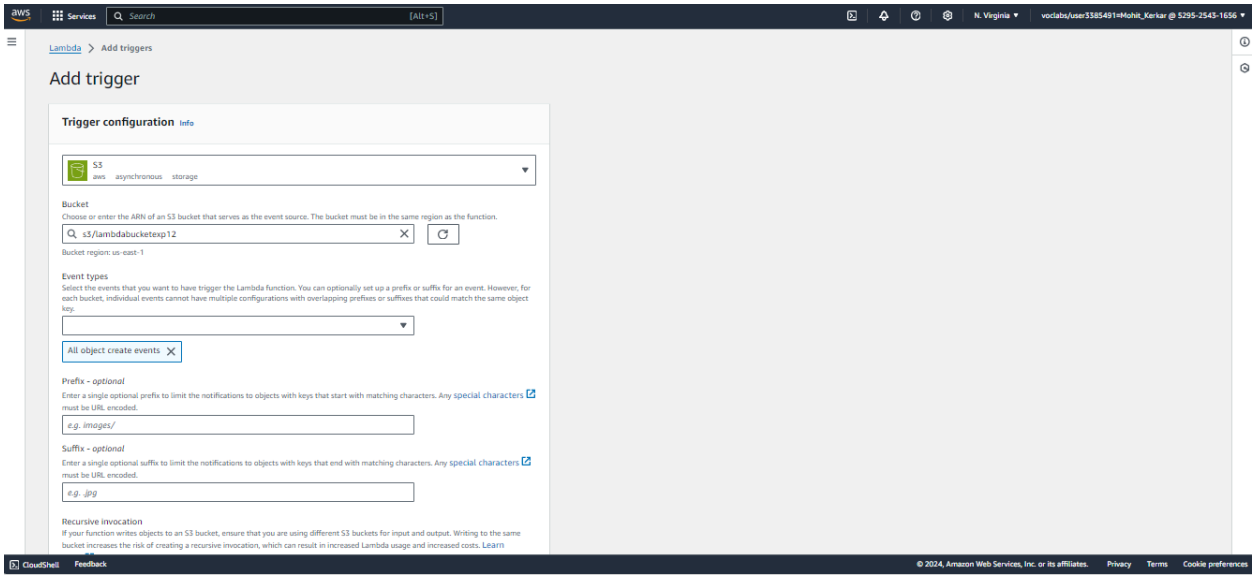


Go to the configuration section inside the function and select Triggers. This is what should appear on the screen…

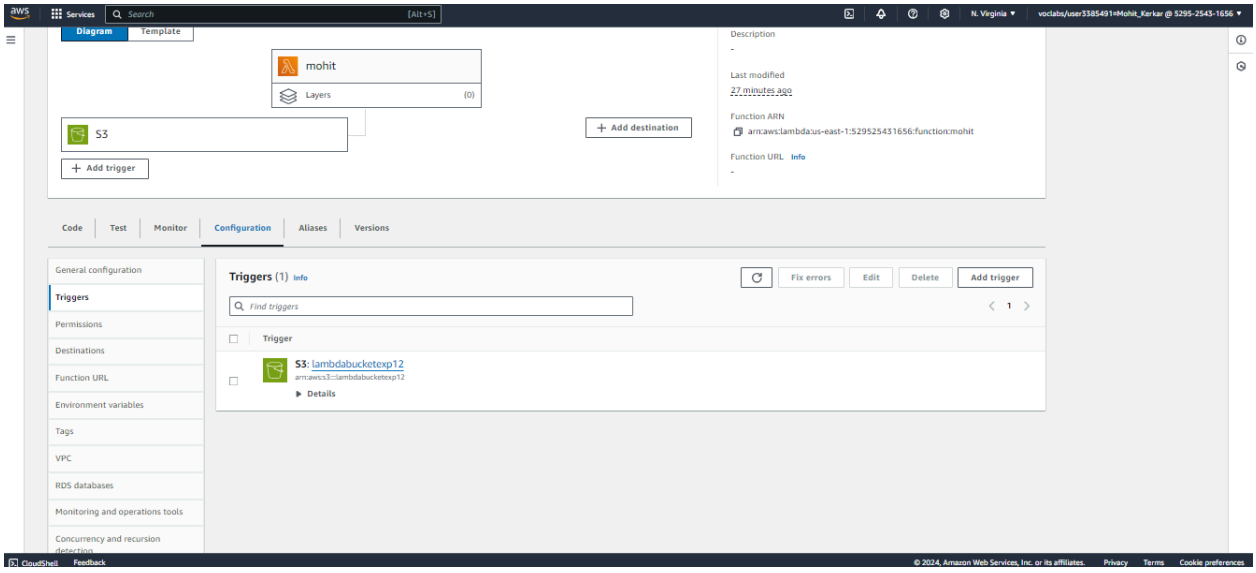


Name: Mohit Kerkar Div: D15C Roll No: 23

Select S3 bucket inside Trigger configuration to use the S3 bucket that was created earlier



After saving the changes, we see those changes being applied in the Configuration section

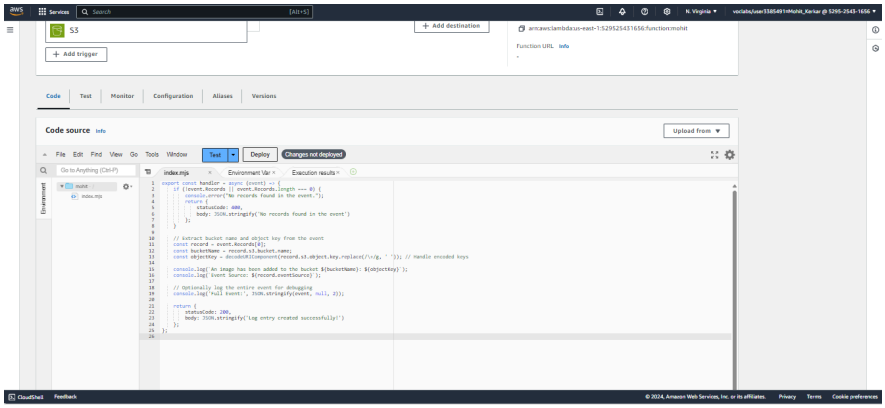


Step 4: Testing our code

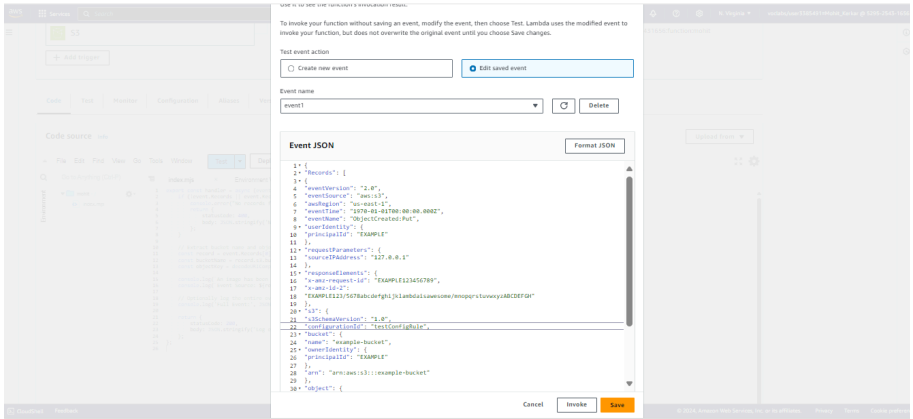
In the code section, we are replace the default code with the following code:

```
export const handler = async (event) => {
  if (!event.Records || event.Records.length === 0) {
    console.error("No records found in the event.");
    return {
      statusCode: 400,
      body: JSON.stringify('No records found in the event')
    };
  }
```

```
    // Extract bucket name and object key from the event
    const record = event.Records[0];
    const bucketName = record.s3.bucket.name;
    const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle
encoded keys

    console.log(`An image has been added to the bucket ${bucketName}: ${objectKey}`);
    console.log(`Event Source: ${record.eventSource}`);

    // Optionally log the entire event for debugging
    console.log('Full Event:', JSON.stringify(event, null, 2));

    return {
        statusCode: 200,
        body: JSON.stringify('Log entry created successfully!')
    };
};
```

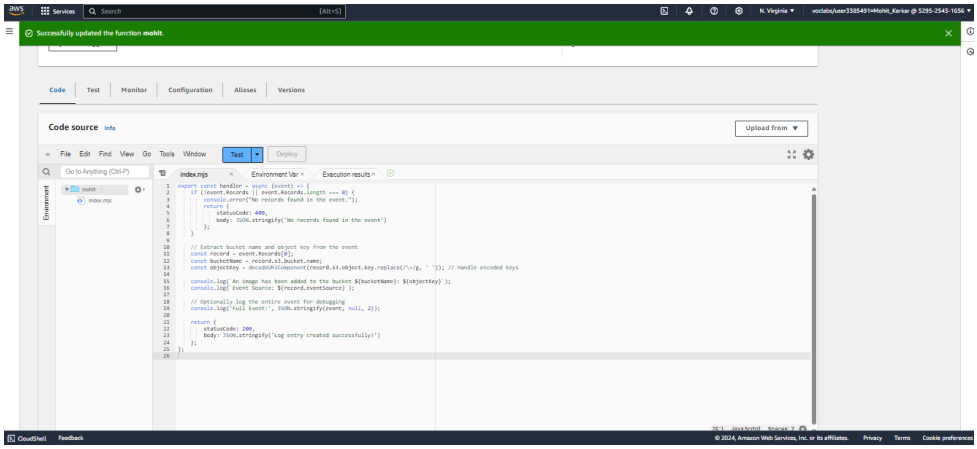


For this particular code, we are supposed to edit the configurations for our test and save the event's JSON as we see in the following screenshot

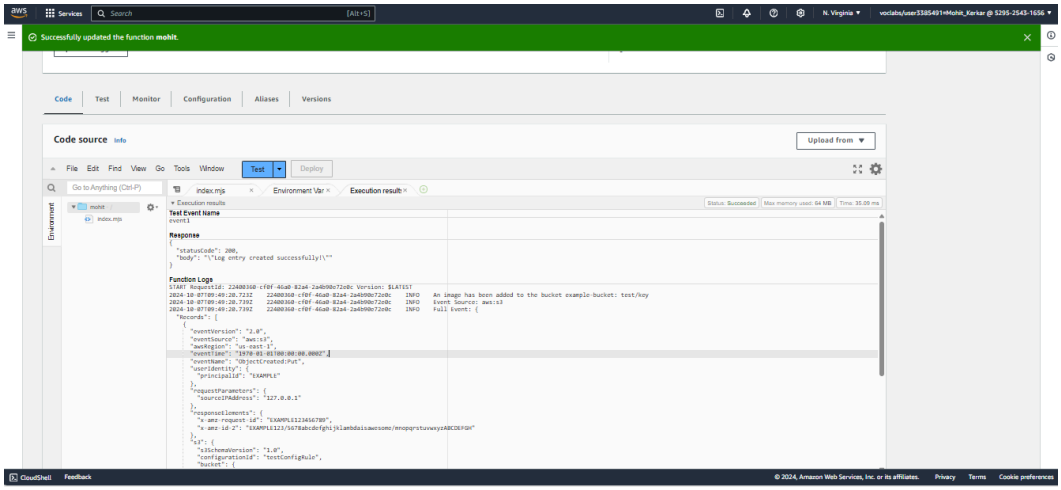


Name: Mohit Kerkar Div: D15C Roll No: 23

Click on deploy

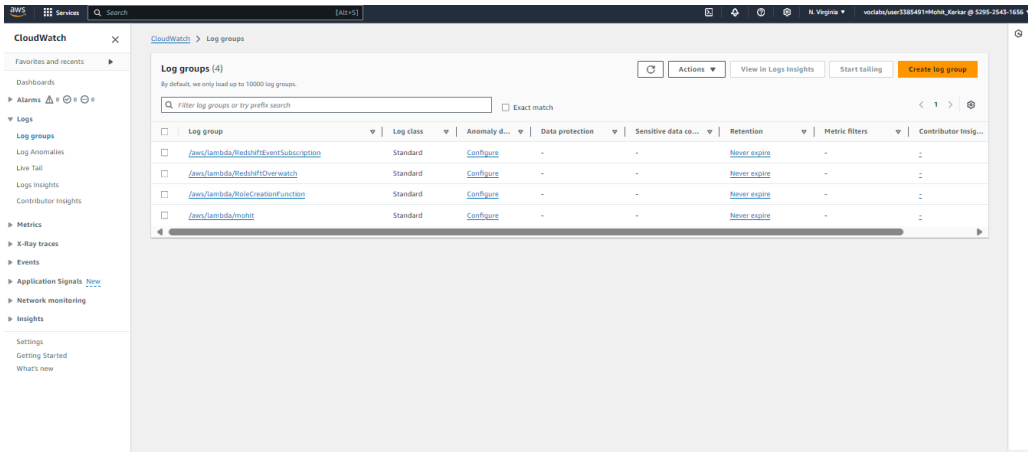


Click on test after deploying. This is the execution result that we see. Here, we see a line being mentioned inside the execution result…. ‘An image has been added to the bucket’.



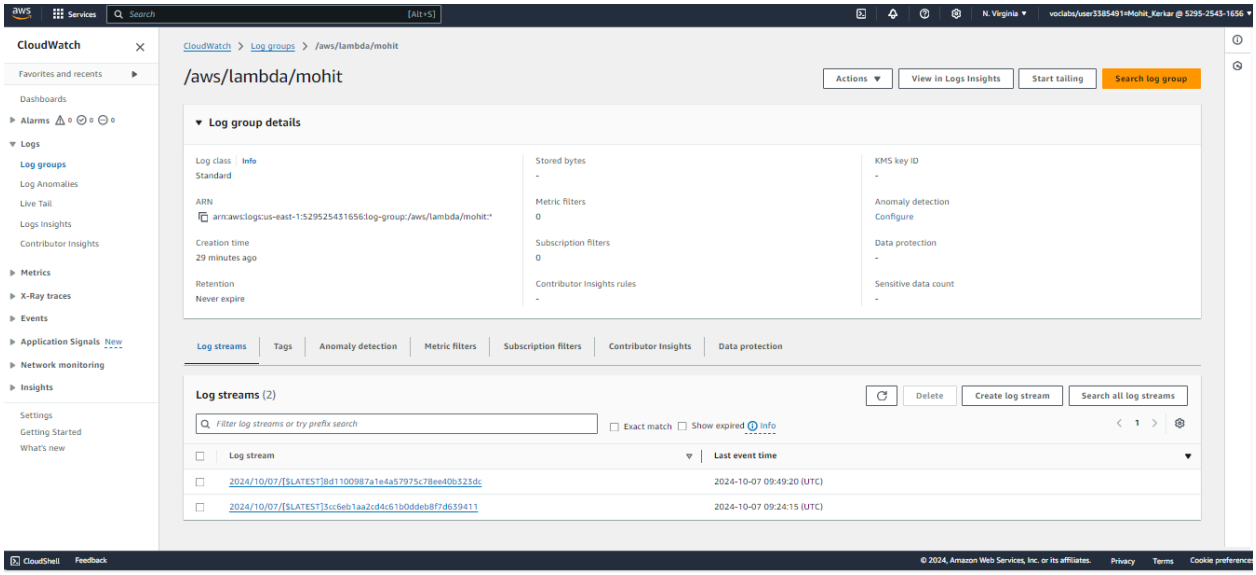
Step 5: Check logs

Search for CloudWatch. Click on logs -> logs groups

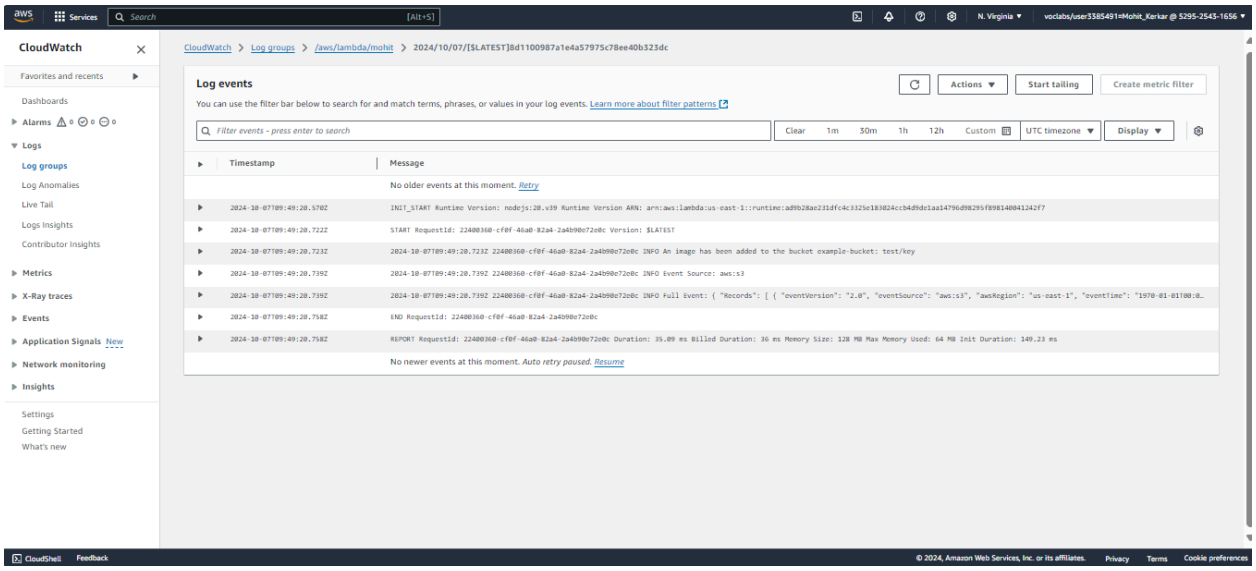


Name: Mohit Kerkar Div: D15C Roll No: 23

In here, select the logs that we want to see



Here, we see it saying ‘An image has been added to bucket’, which matches the execution results in the test that we performed on our code



Conclusion: Integrating AWS Lambda with S3 allows for real-time, automated processing of events such as file uploads. In this example, a Lambda function is configured to log a message whenever an image is added to a specific S3 bucket. This setup demonstrates the power and flexibility of serverless computing by automating tasks without requiring manual intervention or server management. By leveraging AWS Lambda, developers can efficiently handle event-driven workflows, reduce operational overhead, and quickly deploy scalable solutions that respond to specific actions within cloud environments.